Orlando Camargo Rodríguez

Fundamentals of Underwater Acoustics

Springer

Fundamentals of Underwater Acoustics

Orlando Camargo Rodríguez

Fundamentals of Underwater Acoustics

Springer

Orlando Camargo Rodríguez
Department of Physics
University of Algarve
Faro, Portugal

ISBN 978-3-031-31318-9 ISBN 978-3-031-31319-6 (eBook)
https://doi.org/10.1007/978-3-031-31319-6

© The Editor(s) (if applicable) and The Author(s), under exclusive license to Springer Nature Switzerland AG 2023

This work is subject to copyright. All rights are solely and exclusively licensed by the Publisher, whether the whole or part of the material is concerned, specifically the rights of reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.
The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.
The publisher, the authors, and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Cessem do sábio Grego e do Troiano
As navegaçõoes grandes que fizeram;
Cale-se de Alexandro e de Trajano
A fama das vitórias que tiveram;
Que eu canto o peito ilustre Lusitano,
A quem Neptuno e Marte obedeceram:
Cesse tudo o que a Musa antígua canta,
Que outro valor mais alto se alevanta.

Hear no more from the wise Greek
and from the Trojan about all their great
navigations indeed;
Speak no more the fame of Alexander's and
Trajan's all great victories achieved;
Because I sing the illustrious Lusitanian
heart, To whom once bowed Neptune
and Mars:
Hear no more what the old Muse sings,
Because another higher value my song
brings.

Luís de Camões

OS LUSÍADAS

Dedicated to Leo and Lali:
sorry for being absent so often.
Dedicated also to Paulo Felisberto,
who departed too soon.

Preface

Nullius in verba
Royal Society Motto

Young man, in mathematics you don't understand things.
You just get used to them.
John von Neumann

Do we need another book about Underwater Acoustics? Is there is a lack of bibliography on the subject? What about *Principles of Underwater Sound* by Robert J. Urick [1]? or *Fundamentals of Ocean Acoustics* by Brekhovskikh and Lysanov [2]? or the mandatory *Computational Ocean Acoustics* by Jensen, Kuperman, Porter and Schmidt [3]? Those are indeed exceptional references, but in my humble opinion all of them suffer from the same "limitations", namely:

- None of them approach the subject from a unifying perspective; for instance, a physicist feels puzzled with frequent references to the transfer function, while a specialist in signal processing reads impatiently his specific literature, searching for topics related to the impulse response; most probably they are both searching for the same mathematical entity, which is named differently depending on the area of research.
- All the above references neglect the presentation of computational code which, without replacing the essence of the concepts, can be of extreme importance for the development of applications in the short and long term.

Therefore, here it is (without any pretensions) an attempt to fix such limitations: a book on Underwater Acoustics, with a structure which is more organic than logical, unifying diverse areas, and with code fragments written in Python (an open-source programming language) complementing the discussion. This is a book written for beginners and specialists, for biologists, oceanographers, computer engineers, physicists and mathematicians, for civilian and naval personnel. Can this approach be

appropriate? I really hope so, but it is up to the readers to give a positive or a negative answer to this question. Whatever the final judgement is it should be stressed that any errors found in the text (although accidental) are of the exclusive author's responsibility.

To conclude: I would like to thank all my SiPLAB's colleagues for their friendship and direct and indirect contributions to this book: Sérgio Jesus, Paulo Felisberto (which unfortunately is no longer with us), António João Silva and Paulo Santos; to the personnel from MarSensing, Cristiano Soares and Friedrich Zabel. I am particularly grateful to Sérgio for his patience and valuable comments in the treacherous area of signal processing; without his comments I would have induced in error (albeit, involuntarily) many of the book readers. I would like also to name Emanuel Ey, who ported the TRACEO code from Fortran to C and explored its eventual parallelization in the CUDA platform. I extend also my acknowledgements to the personnel from IEAPM, Fernando Marin, Vicente Barroso and specially to Leandro Calado and Rogério Moraes for their unbounded trust in my experience. Rogério's contributions for the parallelization of TRACEO3D and his implementation of the Simplex method for the calculation of eigenrays enriched the model with a huge potential. It would be unfair not to include in the acknowledgements Michael Porter from HLS Research Inc. in San Diego, Frédéric Sturm from Ecole Centrale de Lyon and Pavel Petrov from the Il'ichev Pacific Oceanological Institute in Vladivostok for the privilege of countless interactions.

And having said it all it just remains to wish an enjoyable reading.

Faro, Portugal Orlando Camargo Rodríguez

Contents

1 A Bit of History 1

2 The Ocean 5
2.1 Introduction 5
2.2 Typical Scales 6
2.3 The Deep Ocean 7
2.4 The Shallow Ocean 7

3 Signals and Noise 9
3.1 Introduction 9
3.2 Noise 10
3.2.1 White Noise 10
3.2.2 Correlated Noise 15
3.3 Signals 18
3.3.1 The Unit Impulse 20
3.3.2 The Rectangular Signal 21
3.3.3 Tones 22
3.3.4 Frequency Modulation 26
3.3.5 M-sequences 28
3.3.6 Wavelets 31
3.3.7 The Analytic Signal 31
3.3.8 Signal Filtering 33
3.3.9 Signal Correlation 38
3.4 The Impulse Response 42
3.5 Fourier Synthesis 45
3.6 The Signal-to-Noise Ratio 46
3.6.1 Definition 46
3.6.2 Signal Contamination with Noise 47
3.6.3 Estimating the SNR 48
3.7 Detection Theory 49
3.8 Cepstrum Analysis 54

4 Waves in Boundless Media 57
4.1 Introduction 57
4.2 The Wave Equation 57
4.3 Analytical Solutions 58
4.3.1 Plane Waves 59
4.3.2 Cylindrical Waves 61
4.3.3 Spherical Waves 63
4.3.4 Sommerfeld's Identity 66
4.4 Phase Velocity, Group Velocity 66
4.5 Attenuation 69
4.6 The Doppler Effect 73

5 Waves in Bounded Media 79
5.1 Introduction 79
5.2 The Navier–Stokes Equation 79
5.3 Acoustic Impedance 85
5.4 Wave Dispersion 86
5.5 Wave Refraction 87
5.6 The Speed of Sound 89
5.7 Wave Reflection 90
5.8 Beam Displacement 97
5.9 Final Remarks 100

6 The SONAR Equation 101
6.1 Introduction 101
6.2 The Passive SONAR Equation 102
6.3 The Active SONAR Equation 102
6.4 The Signal Excess 102
6.5 The Source Sound Level 103
6.6 The Transmission Loss 103
6.7 The Noise Level 105
6.8 The Directivity Index 105
6.9 The Target Strength 106
6.10 The Reverberation Level 107
6.11 Examples 108

7 Arrays, Transducers and Vector Sensors 109
7.1 Introduction 109
7.2 Array with Two Transducers 109
7.3 Array with Several Transducers 112
7.4 Linear Array 112
7.5 Disk Array 115
7.6 Frequency Response 115
7.7 Vector Sensors 116

8 Propagation Models 119
8.1 Introduction 119
8.2 Ray Tracing 120
8.2.1 Ray Trajectories 120
8.2.2 The Hamiltonian 128
8.2.3 Geodesics 130
8.2.4 Ray Amplitudes 133
8.2.5 Ray Interference 135
8.2.6 Source Directivity 137
8.3 Normal Modes 137
8.3.1 The Isovelocity Waveguide 141
8.3.2 The Pekeris Waveguide 146
8.3.3 Airy Phase 150
8.3.4 Adiabatic Propagation 151
8.3.5 Modes and the Reflection Coefficient 151
8.3.6 The Waveguide Invariant 153
8.3.7 Numerical Calculation of Modes 157
8.3.8 Source Directivity 162
8.4 The WKBJ Approximation 162
8.5 Fast-Field Models 164
8.6 The Parabolic Equation 169

9 Scattering and Reverberation 175
9.1 Introduction 175
9.2 Lambert's Model 177
9.3 Surface Scattering 177
9.4 Surface Backscattering 177
9.5 Bottom Scattering 182
9.6 Multibeam and Sidescan Sonars 188
9.7 Boundary Reverberation 189
9.8 Example 189

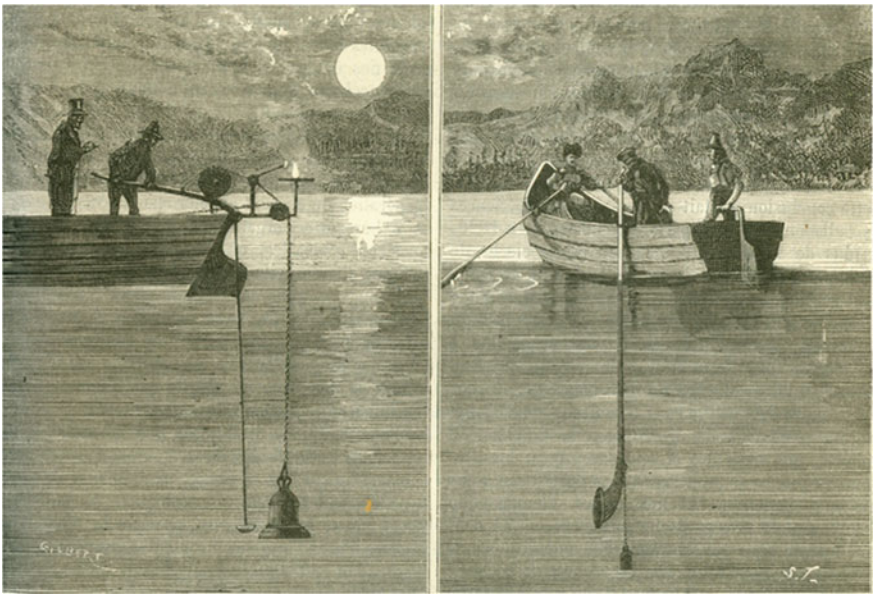
10 Inverse Problems 193
10.1 Introduction 193
10.2 Source Localization 194
10.2.1 Bearing Estimation 194
10.2.2 Range and Depth Estimation 197
10.3 Geoacoustics 202
10.4 Tomography 205
10.4.1 Matched-Field Tomography 206
10.4.2 Travel-Time Tomography 207

Appendix A: Sea Surface Roughness 209

Appendix B: Solving a System of Linear Equations 215

References 219

Chapter 1
A Bit of History



The experiment at Lake Geneva

(*image in the public domain*)

It seems appropriate to place the official birth of Underwater Acoustics at the year 1826, when the Swiss physicist Jean-Daniel Colladon performed a memorable experiment at Lake Geneva: at night, and in the company of the French mathematician Jacques Charles François Sturm, which was on another boat at a distance of 10 km, Colladon repeatedly knocked on a bell located underwater; at the same time he could also produce a brilliant spark by igniting a small amount of powder. On the receiving side, and with the help of an underwater tube located at the same depth of the bell,

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023

O. C. Rodríguez, *Fundamentals of Underwater Acoustics*,
https://doi.org/10.1007/978-3-031-31319-6_1

Sturm could measure the time required for the bell sound to be heard, by starting the time count from the moment the spark was visible. The measured speed of sound was 1435 m m/s in a water at a temperature of 8 °C; this value differs from more accurate measurements achieved with modern technology just by 3 m/s. The particularities of underwater sound propagation were in fact known long before Colladon and Sturm experiment. Leonardo da Vinci, for instance, described in 1490 a simple device that can be considered as the precursor of what is known today as a passive sonar system; Leonardo wrote that "*While your ship stands still on the surface of the water, place a long tube underwater and listen carefully to the upper side: you will be able to hear vessels, which are still not visible*". More than a century later, in 1620, Marin Mersenne discovered a mathematical description of the law of vibrating strings, and in 1687 Sir Isaac Newton presented the first mathematical theory of the propagation of sound in the air, which was published in the *Philosophiae Naturalis Principia Mathematica*. The experiment at Lake Geneva took place in a moment which in certain sense defined a past and a future in Underwater Acoustics, let it be as a source of inspiration for different applications, or as a reference point for further theoretical developments. Thus, Charles Bonnycastle developed in 1838 the first experiment for the calculation of water depth, based on echo ranging; he was followed by Matthew Fontaine Maury (at the time director of the American Repository of Navigation Charts and Instruments), who tried unsuccessfully in 1859 to measure bottom depth. From 1877 to 1878 a certain John William Strut (who would become later known as *Lord Rayleigh*) published *The Theory of Sound*, a paramount work in two volumes, who introduced for the first time the wave equation, and established the foundations for the development of research in Underwater Acoustics and its applications in the twentieth century.

It seems appropriate to take a temporary detour at this point and notice that, since its origins, the development of underwater vehicles had been strongly related with research in Underwater Acoustics [4]. When in 1800 the American Robert Fulton showcased the *Nautilus* to the French Republic (and the vessel can be indeed considered as the first of its kind), many saw it as the ideal tool to compensate the power of the Royal Navy; the submarine, nevertheless, was never used in action.[1] The potential of submarines would be to be demonstrated later in a tragic episode of the American Civil War, when the confederate submarine *H. L. Hunley* managed to sink the Union ship *USS Housatonic*, by detonating underneath of it an explosive charge, which was located at the extreme of a harpoon. The Hunley sunk immediately after the explosion, apparently due to crew poisoning with carbon dioxide (the air was supposed to be renovated every 30 min) or injured by the underwater shock wave. The events related to the Hunley were followed with great interest in Europe. In 1879 the British developed the *Resurgam*, the first submarine with engine propulsion; it was followed by the French *Goubet I* in 1885, by the British *Nautilus*, with electric engines, in 1886 (what a lack of imagination!) and by the Spanish *Peral* in 1888, and *Ictineo* in 1889. In such context it was a matter of time before the problem of underwater source localization started to become of extreme relevance. The first

[1] Fulton, as a *entrepreneur*, did not have any problems in showcasing the Nautilus to the British, who lost interest on it after the battle of Trafalgar.

technological development of interest was presented in 1901 by the *Submarine Signal Company*, who decided to install underwater bells near navigation beacons, which could be heard by hydrophones located on every vessel. It was indeed an original idea, which failed completely because the hydrophones were able only to hear the noise produced by the vessel engines or the one generated by the chaotic motion of the water while the vessel was moving. Instead of giving up, the company decided to hire in 1912 the Canadian engineer Reginald A. Fessenden to modify the hydrophones and filter the noise. Fessenden, instead, suggested to replace the bells with acoustic sources, which would transmit messages in Morse code. With the disaster of the *Titanic* Fessenden shifted his attention to the development of a mechanic transducer, which could be used for *echo ranging* with underwater signals. His idea was tested successfully in 1914, when Fessenden managed to detect an iceberg at a distance of 2 miles ($\approx$3 km); additionally, he also managed to detect the ocean bottom at a depth of 186 feet ($\approx$57 m). Fessenden's transducer would end up to be replaced with crystal transducers, which work on the basis of the *piezoelectric* effect: an electric current applied to a crystal generates a mechanic vibration, which can be used to produce an acoustic signal; inversely, acoustic signals can induce the vibration of a crystal, which generates an electric current. In fact, the first functional sonar system with a crystal transducer was developed by the French physicist Paul Langevin during World War I in 1917, motivated by the need of finding the German *U-boot*s, which relentlessly harassed the commercial routes of the Allies. World War II and the Cold War provided additional developments of extreme importance, in particular, thanks to the advances in signal processing and computing power, which progressively would be concretized in the form of numerical propagation models. Of extreme relevance was the discovery of the ocean *SOFAR* channel; its behaviour at different latitudes was at the base of the *SOSUS* system which, during the Cuban missile crisis, allowed the American Navy to track several Soviet submarines. Classified information about the SOSUS would become known to the Soviets in 1988, leading to the development of the *Akula* ("Shark", in Russian) submarines, which no longer could be detected by the SOSUS system due to their reduced acoustic signature.

The historical references are too numerous to keep discussing them. Research in Underwater Acoustics continues, with the support of computational resources which become more and more powerful every day. Underwater communication based on the transmission and reception of neutrinos is mentioned from time to time as the final solution to the problem of submarine communication, and Artificial Intelligence is referred to as a perfect tool for many applications, such as vessel classification, source localization and even to decode the language of whales. Yet, the history of Underwater Acoustics is very likely to remain constantly unfinished. Whatever the advances in transducer technology, signal processing or computational power, the fact is that Underwater Acoustics still remains a formidable challenge. By way of example it is worth to mention an incident that happened in 2009, when somewhere in the North Atlantic the French nuclear submarine *Le Triomphant* collided accidentally with the British nuclear submarine *H.M.S. Vanguard*. Such collision, which took place despite the sophisticated technology available at each vessel, is particularly enlightening of how complex the problems of Underwater Acoustics can be. These problems indeed deserve our deepest attention.

Chapter 2
The Ocean

2.1 Introduction

The ocean is, beyond any doubt, the main protagonist of this book, the immense stage, exceptionally dynamic and complex, which only Underwater Acoustics can sense in an efficient way at small and large scale; even so it is surprising how little we know about it and the little esteem with which it is regarded. Let us notice, for instance, that

- Three quarters of the planet are covered by the ocean waters; it is home to millions of animal and vegetal species, some of which thrive in environmental conditions which at a first sight seem adverse to the existence of living organisms.
- Global currents circulate between the continental coasts, with climate effects of which a common citizen is not aware. For instance: Lisbon and New York are located approximately at the same geographical latitude; however, in the winter season, New York has to endure intense snowfalls and sub-zero temperatures, while at the same time Lisbon experiences periods of rain and light cold, with positive temperatures. This is only possible because in the winter season the thermohaline current transports masses of warm water along the Portuguese coast, while at the same time cold water is transported by the current along the coast of New York.
- Ocean currents are responsible for the *upwelling* phenomena, which are so beneficial to the fishing industries of Portugal, Brazil or South Africa.
- Hidden from view the ocean harbours active volcanoes and tectonic plates, which at any moment can trigger devastating tsunamis.
- The ocean is responsible for the mitigation of the greenhouse effect by capturing carbon dioxide, while at the same time it functions as a global road for the trade of goods and merchandise and provides support for the feeding of millions of people. Even so the ocean's exploitation seems to know no boundaries, and the contamination of the oceans seems unstoppable; some estimates indicate that in one or two decades there will be more plastic than fish in the ocean.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023

O. C. Rodríguez, *Fundamentals of Underwater Acoustics*,
https://doi.org/10.1007/978-3-031-31319-6_2

The physical ocean is studied in detail in the context of *Oceanography*, which is based on the mathematical formalism of Physics applied to the motion of water masses worldwide. However, within the context of Underwater Acoustics, many concepts of Oceanography are to be adapted depending on the applications, mostly by considering a model of the ocean in which:

- The complex and certainly important distribution of marine organisms is extremely simplified, by considering only physical parameters of the water column, such as pressure, temperature and salinity. In several applications marine organisms are to be considered in the role of acoustic sources; the same principle can be adopted for surface and/or underwater vessels and to the geologic activity.
- Whatever the environment being considered the methods of signal processing and wave propagation are to be taken into account, conditioned by the particularities of the speed of sound, and in combination with the specific properties of the surface and the bottom.

In order to optimize this model the ocean is further differentiated according to the typical scales of oceanic phenomena, and by introducing a formal division in terms of deep and shallow oceans; the scales and this formal division are to be addressed in the next sections.

2.2 Typical Scales

While there is no consensus regarding this issue there are three space-time scales, which are often mentioned in the literature of this specialty; the scales are presented in Tables 2.1 and 2.2, together with the associated phenomena [5]. Generally, the bigger the spatial scale of a given ocean phenomenon, the bigger should be the wavelength considered for its corresponding monitoring.

Table 2.1 Ocean space-time scales

Denomination	Domain	Typical length	Typical duration
Global	Planetary	>100 km	Season
Mesoscale	Continental	100 m–100 km	Weeks
Fine scale	Coastal zone	<100 m	Hours

Table 2.2 Scales and associated phenomena

Scale	Associated phenomena
Global	Thermohaline circulation, El Niño, La Niña
Mesoscale	Currents, fronts and vortexes, internal tides
Fine	Thermohaline staircases, Langmuir circulation

2.3 The Deep Ocean

The deep ocean can be defined as the part of the ocean in which bottom depth is above 200 m [6]. The corresponding distributions of water salinity and temperature change from place to place, but in many cases they create conditions of horizontal stratification, which enhance the propagation of acoustic signals at very long distances, without reflections on the surface and the bottom. Such stratification is disturbed by fronts, vortexes and currents, leading to complex regimes of three-dimensional acoustic propagation; the existence of mesoscale phenomena is not evident at the surface of the ocean; this is why acoustic monitoring is the most efficient way to infer the spatial distributions of salinity and temperature. In some situations the depth interval between 700 and 1000 m can exhibit the presence of a *thermocline*, which is a sudden variation of temperature, induced by the superposition of two different water layers, with radically different properties. The existence of underwater canyons can also manifest itself in complex patterns of directionality of the transmitted signals. Additionally, and independently of the environment being considered, the progressive increase of water pressure with depth becomes the dominant factor of signal propagation, with the trend being mirrored in the dependence of sound speed on water depth.

2.4 The Shallow Ocean

The shallow ocean can be defined formally as the locations in which bottom depth is below 200 m [6]. While this feature of the ocean channel favours the propagation of acoustic signals with "high" frequencies it also makes more relevant the problem of boundary reflections; both the surface and the bottom can be characterized by drastically different sets of physical properties and roughness. In the case of the continental platform the reflection of a signal on a sloped boundary can induce the creation of acoustic "shadows", which impose limitations on the performance of systems for the detection of acoustic sources. In particular, and as a result of its interaction with the bottom, the acoustic signal will be attenuated after each reflection, which at a first sight could be considered as a disadvantage; however, such interaction can in fact provide an efficient method to determine the bottom properties replacing the cumbersome task of direct bottom sampling.

Chapter 3
Signals and Noise

3.1 Introduction

Perhaps it might seem rather unorthodox to start the discussion of Underwater Acoustics talking about signals and noise; however, this book has been designed with a structure which is more organic than logical, and it is a fact that without signals and noise there would not be discussions to start with. Inevitably, such discussion will require concepts of signal processing, which are systematically mentioned in the literature of this book's subject; these concepts are here grouped initially in terms of *signals* and of *noise*, and later in terms of signals and noise, in order to define the concept of the *signal-to-noise ratio*, which stands at the base of the SONAR equation. Before proceeding the reader should keep in mind that:

- Although the discussion is presented in a rather general context one must not forget that we are considering *acoustic* signals, that is, signals related to the propagation of mechanical waves in a continuous medium, in the frequency interval 20 Hz and 20 kHz; signals associated with mechanical waves with frequencies above 20 kHz are called *ultrasound*, while signals associated with mechanical waves with frequencies 20 Hz are called *infrasound*.
- Signal processing can be the processing of *analog* or *digital* signals. In the first case time can be considered as a continuous variable, and the signal can be known at any instant t; in *Digital Signal Processing* (abbreviated as DSP) it is considered that the signal is sampled regularly at particular instants of time, and this raises important questions related to the proper sampling of the signal. However, since this is book of *fundamentals* and not a book on signal processing, and since there is a commitment to complement the discussion with numerical examples, it was decided to allow some freedom in the presentation of the topics, which will be presented from a perspective of analog signals, but will be showcased in code as often done when DSP is being considered.
- The definitions of noise and signals presented in the following sections are far from perfect; however, following the main philosophy of this book, it is believed that the specific concepts here presented will ease the assimilation of more general

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023

O. C. Rodríguez, *Fundamentals of Underwater Acoustics*,
https://doi.org/10.1007/978-3-031-31319-6_3

topics often found in the typical literature of Underwater Acoustics. This approach to the discussion can be adopted, extended or modified, according to the specific needs of the reader.

3.2 Noise

We will define the undesired component of a reception as *noise*; noise, in general, is a contamination of the reception, it is independent of the source of interest and in general it has a random nature; under such perspective noise stands as a stochastic process[1]. The randomness does not imply that there is a unique type of noise; in order to simplify matters we will consider that there are two fundamentals types of noise, namely, *white* noise and *correlated* noise.

3.2.1 White Noise

White noise can be defined as a sequence of random values, perfectly uncorrelated in time, or having a spectral density which is constant over frequencies. The denomination "white" can be justified through an analogy with Optics, in which the combination of different colours of the visible spectrum in equal proportions produces white light. White noise can be of different types depending on the distribution of random variables in a stochastic process. In the sections that follow we will consider only two types of white noise, namely, *uniform* white noise and *Gaussian* white noise.

Uniform White Noise

Uniform white noise is the noise that has a constant *probability density function* (abbreviated PDF) given by the expression [7]

$$f(u) = \frac{1}{\beta - \alpha} \tag{3.1}$$

in which $\alpha \leq u \leq \beta$. Additionally, the corresponding *cumulative distribution function* (abbreviated as CDF) corresponds to

$$F(u) = \begin{cases} 0 & u < \alpha \\ \dfrac{u - \alpha}{\beta - \alpha} & \alpha \leq u \leq \beta \\ 0 & u > \beta \end{cases} \tag{3.2}$$

By the general definition of mean value one can find, unsurprisingly, that the mean of a sequence of uniform random values corresponds to

[1] In probability theory, by convention, a *variable* is random, but a *process* is stochastic.

$$\mu = E[u] = \int_{-\infty}^{\infty} u\ f(u)\ du = \frac{\alpha + \beta}{2} \tag{3.3}$$

It is important to notice that in the case of a discrete and finite sampling of uniform random values what it is implemented as the mean value coincides with the arithmetic mean:

$$E[u] = \frac{1}{N}\sum_{n=0}^{N-1} u_i$$

The theoretical discussion is here complemented with the Python Code Snippet 3.1, which exemplifies the numerical generation of 1001 samples of uniform white noise, for the particular case in which $\alpha = -1$ and $\beta = 1$; according to Eq. (3.1) it follows that

$$f(u) = \frac{1}{1 - (-1)} = 0.5$$

In order to validate the results the code produces a normalized histogram of the samples, with the data grouped in 20 columns (since the histogram is the numerical equivalent of a PDF it is often called the "empirical probability density"). The histogram of Fig. 3.1, as expected, exhibits a sequence of columns with similar heights close to the value 0.5.[2] The reader is invited to run the code and plot the histogram, and then to "play" with the code by changing the number of samples, of α and β, and in this manner generate different sequences of uniform white noise.

Code Snippet 3.1 Generating uniform white noise.

```
from numpy import *
from matplotlib.pyplot import *

a = -1.0
b = 1.0
n = 1001
u = random.uniform(a,b,n)

figure(1)
hist(u,bins=20,density=True,color='w',ec='k')
show()
```

[2] In a typical histogram it is defined a number of N_b columns, and the interval between min X and max X of the N values of X is divided into N_b segments, each with a length given by $\Delta X = (\max X - \min X)/N_b$; the height of the ith column represents the number of N_i random variables contained inside the segment in which the column stands; obviously $N_1 + N_2 + N_3 + \ldots + N_b = N$. In a *normalized* histogram the height of the ith column is given by $f_i = N_i/(N\Delta X)$, and thus $f_1\Delta X + f_2\Delta X + \ldots = 1$.

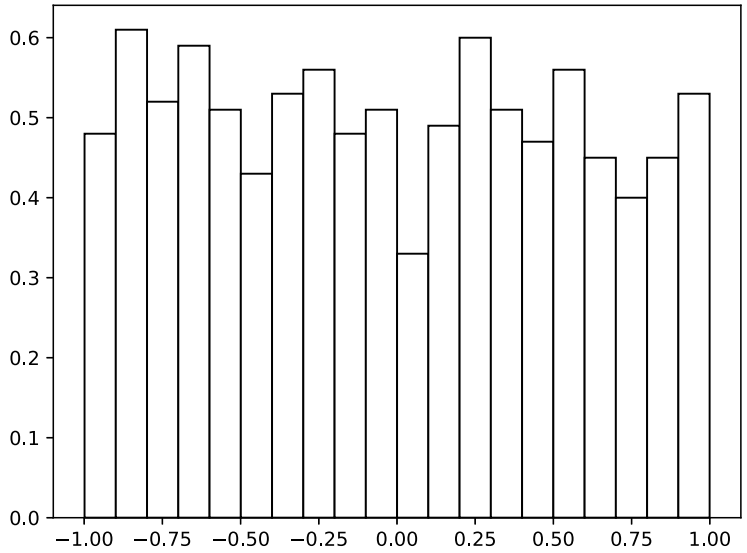


Fig. 3.1 Normalized histogram of a distribution of uniform white noise, with $\alpha = -1$ and $\beta = 1$

Gaussian White Noise

Gaussian white noise with a mean μ and a variance σ^2 is noise with a PDF that corresponds to [7]

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right] \tag{3.4}$$

where $-\infty < x < \infty$; the CDF is given by

$$F(x) = \frac{1}{\sqrt{2\pi}\sigma} \int_{-\infty}^{x} \exp\left[-\frac{1}{2}\left(\frac{u-\mu}{\sigma}\right)^2\right] du \tag{3.5}$$

A common notation to indicate that a random variable X is Gaussian (or *normal*) consists in writing that

$$X \sim N(\mu, \sigma^2)$$

As in the previous case the discussion is here complemented with the Python Code Snippet 3.2, which exemplifies the generation of 1001 samples of Gaussian white noise in the particular case with $\mu = 0$ and $\sigma = 1$; according to Eq. (3.4) it follows that $f(0) = 1/\sqrt{2\pi} \approx 0.4$. Again, the code generates a normalized histogram with 20 columns, which is shown in Fig. 3.2; as expected, the histogram is symmetrical around the point $x = 0$, with a maximal height of 0.4, and with the column height decreasing progressively around that point as given by Eq. (3.4); under these conditions the

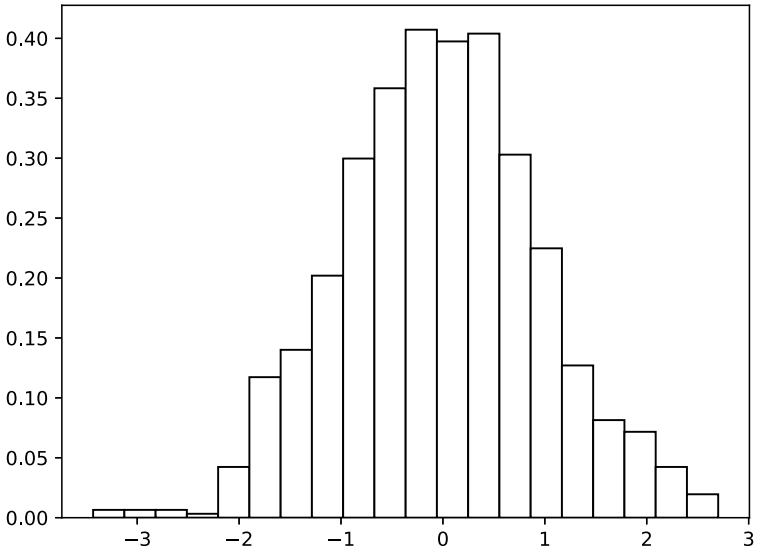


Fig. 3.2 Normalized histogram of a distribution of Gaussian white noise, with $\mu = 0$ and $\sigma = 1$

histogram allows to validate that the numerical values indeed represent Gaussian white noise.

Code Snippet 3.2 Generating Gaussian white noise.

```
from numpy import *
from matplotlib.pyplot import *

mu, sigma = 0, 1.0
n = 1001
g = random.normal(mu, sigma, n)

figure(1)
hist(g,bins=20,density=True,color='w',ec='k')
show()
```

Converting Gaussian Noise to Uniform Noise

Let us illustrate an important application of the CDF, by converting Gaussian white noise to uniform white noise. Let be $X \sim N(\mu, \sigma^2)$ and F_X the corresponding CDF (see Eq. (3.5)); it can be shown that the random variable Y defined as

$$Y = AF_X(X/\sigma) - B$$

will have an uniform distribution in the interval $[-B, A - B]$. This application of the CDF is illustrated in the Python Code Snippet 3.3, which shows, in particular, the conversion of a Gaussian random variable with $\mu = 0$ and $\sigma = 1$ to a uniform random variable distributed in the interval $[-5, 5]$, which corresponds to $A = 10$ and $B = 5$.

Code Snippet 3.3 Conversion of Gaussian white noise to uniform white noise.

```
import matplotlib.pyplot as plt
import numpy as np
import scipy.stats as stats

# Gaussian white noise:
mu, sigma = 0, 1
g = np.random.normal(mu, sigma, 1001)
# Converting to uniform:
A, B = 10, 5
# Use the CDF to get uniform white noise:
u = A*stats.norm.cdf(g/sigma, mu, sigma) - B
plt.figure(1)
plt.hist(g,bins=20,density=True)
plt.figure(2)
plt.hist(u,bins=20,density=True)
plt.show()
```

Complex Gaussian White Noise

It might be the case that a particular application requires the generation of *complex* Gaussian white noise, and care should be taken to avoid generating noise samples of the type $n_i = r_i + is_i$, where r_i and s_i correspond to real random variables, generated as explained in the previous section. The complex noise generated in such a way will be, in fact, *correlated*, in the sense that its real and complex parts will not be statistically independent. The generation of truly complex Gaussian white noise with a zero mean and a variance σ can be achieved using the *Box–Muller algorithm*, in which the complex samples are to be generated using the expression [8]:

$$n_i = \sigma\sqrt{-\log X_i}e^{i2\pi Y_i} \tag{3.6}$$

where X_i and Y_i are real random variables, with uniform distributions in the interval $(0, 1]$. In order to obtain complex noise with a mean different from zero it is enough (as done in the Python Code Snippet 3.2) to add the desired mean value to the samples generated by Eq. (3.6).

3.2.2 Correlated Noise

Correlated noise is noise which is non-white; it can be even of the specific "colour", depending on the distribution of energy in the frequency domain. For instance: "pink" noise (of extreme importance in biological systems) exhibits a distribution of energy which is inversely proportional to the frequency and, according to some authors, can be differentiated from white noise by humans. Pink noise has a distribution of energy which enhances low frequencies in detriment of high frequencies; such behaviour defines the noise "colour" (in the case of light the colour pink is the component of white light, which is located at the lower frequencies of the visible spectrum). White noise can be converted into noise of a specific colour through *filtering* techniques, in which the filter is designed to achieve the desired modulation of frequencies. At his stage of the discussion it would be too soon to provide a specific example, but the conversion of white noise to pink noise will be addressed in a more complete context in Sects. 3.3 and 3.3.8.

Let us consider now the realizations not of a random variable, but of a random *vector*, which will be here represented as

$$\mathbf{X}(t) = [X_1(t), X_2(t), X_3(t), \ldots X_N(t)]^T$$

in other words, $\mathbf{X}$ is a function of time; the index T means transposition. This vector can be characterized by a *multivariate probability distribution* $f(\mathbf{x})$, such that

$$\int_{\mathbb{R}_N} f(\mathbf{x})\, d\mathbf{x} = 1 \tag{3.7}$$

Associated to $\mathbf{X}$ there is a vector mean $\boldsymbol{\mu}$, in which each component is given by

$$\mu_i = E[X_i] \tag{3.8}$$

Besides $\boldsymbol{\mu}$ one can also define the matrices of *correlation* $\mathbf{R}$ and *covariance* $\mathbf{C}$, with components defined by

$$R_{ij} = E\left[X_i X_j\right] \quad \text{and} \quad C_{ij} = E\left[(X_i - \mu_i)\left(X_j - \mu_j\right)\right] \tag{3.9}$$

or in matrix notation

$$\mathbf{R} = E\left[\mathbf{X}\mathbf{X}^T\right] \quad \text{and} \quad \mathbf{C} = E\left[(\mathbf{X} - \boldsymbol{\mu})\,(\mathbf{X} - \boldsymbol{\mu})^T\right]$$

It is possible to demonstrate that

$$\mathbf{C} = \mathbf{R} - \boldsymbol{\mu}\boldsymbol{\mu}^T$$

In the particular case in which $\boldsymbol{\mu} = \mathbf{0}$ the above matrices coincide with each other. The *variance* of $\mathbf{X}$ corresponds to the diagonal elements of $\mathbf{C}$:

$$\mathrm{Var}\left[X_i\right] = C_{ii} \tag{3.10}$$

For the case of a Gaussian vector one can write that [7]

$$f(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^N \det \mathbf{C}}} \exp\left[-\frac{1}{2}\left((\mathbf{x}-\boldsymbol{\mu})^T \mathbf{C}^{-1} (\mathbf{x}-\boldsymbol{\mu})\right)\right] \tag{3.11}$$

In order to indicate that a vector $\mathbf{X}$ has a Gaussian distribution it is often written that

$$\mathbf{X} \sim N(\boldsymbol{\mu}, \mathbf{C})$$

It is worth noting that if $\mathbf{X}$ has a Gaussian distribution, then the vector $\mathbf{Y} = \mathbf{AX} + \mathbf{B}$ also has a Gaussian distribution, with mean $\mathbf{A}\boldsymbol{\mu} + \mathbf{B}$ and with a covariance matrix given by $\mathbf{C}' = \mathbf{A}^T\mathbf{CA}$. The calculation of realizations of a vector $\mathbf{Y}$ for a given covariance matrix is illustrated in the Python Code Snippet 3.3, in which a Cholesky decomposition is used to generate $\mathbf{Y}$ from a Gaussian uncorrelated vector $\mathbf{X}$. In order to verify the correlation of the data the code generates also a pair of figures, showing the original covariance matrix and the estimated covariance matrix (see Fig. 3.3).

Code Snippet 3.4 Generating correlated noise.

```
from numpy import *
from matplotlib.pyplot import *

# Generating correlated noise:
# X => gaussian vector with N = 20
# Covariance matrix:
# E[Xi Xj] = e^|i-j|/5, i,j = 1,...,20.
# Cholesky decomposition => Y = B X

rcParams.update({"text.usetex": True})

N = 20
I = range(1,N+1)
n = 1000
X = random.randn(N,n)
Y = zeros((N,n))
C = zeros((N,N))

for i in range(N):
    for j in range(N):
        C[i,j] = exp(-abs(i-j)/5)

B = linalg.cholesky(C).T

for i in range(n):
```

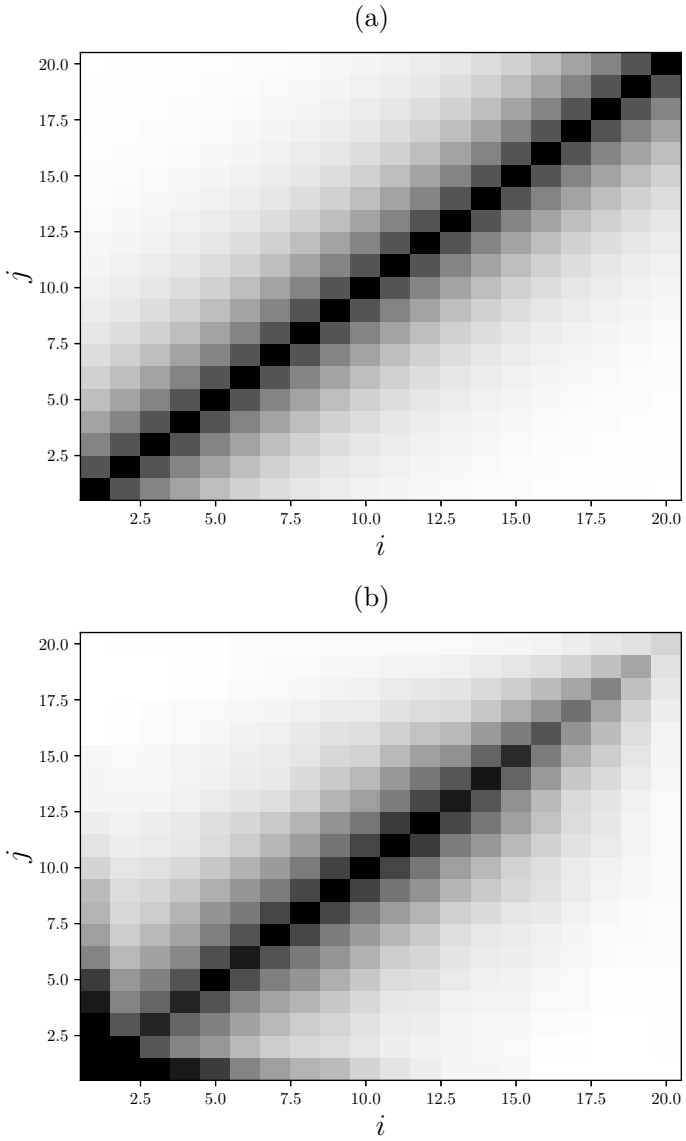


Fig. 3.3 Correlated Gaussian noise: **(a)** original covariance matrix; **(b)** estimated covariance matrix

```
        Y[:,i] = B.dot( X[:,i] )

    Ce = cov( Y )

    figure(1)
    pcolormesh(arange(1,N+1),arange(1,N+1),C,
    cmap='Greys',shading='auto')
    xlabel(r'$i$',fontsize=18)
    ylabel(r'$j$',fontsize=18)

    figure(2)
    pcolormesh(arange(1,N+1),arange(1,N+1),Ce,
    cmap='Greys',shading='nearest',vmin=0,vmax=1)
    xlabel(r'$i$',fontsize=18)
    ylabel(r'$j$',fontsize=18)

    show()
```

3.3 Signals

A *signal*, in opposition to noise, corresponds to the desired component of reception. In a similar way, in opposition to random, a signal can be expected to have a "deterministic" nature in the sense that it fills a specific frequency band over time. Before proceeding with the discussion it is important to define some concepts, which will be used systematically along this chapter, namely:

- Whatever the signal in question it will be considered that in each second it is sampled at N discrete instants of time. The value of N, in Hz, corresponds to the *sampling frequency* f_s.
- Half of the sampling frequency corresponds to the *Nyquist frequency*: $f_N = f_s/2$. Let f_{max} be the largest frequency contained in a sampled signal; the samples of the signal will represent the original signal with a minimal loss of information under the condition that $f_N > f_{max}$; otherwise the signal will suffer *aliasing*, in other words, it will be sub-sampled.
- The quantity $\Delta t = 1/f_s$ represents the *sampling interval*, that is, the temporal separation between a given sample and the next.
- The *Fourier transform* is a tool of signal processing that allows to represent a deterministic signal $s(t)$ in the frequency domain according to the integral representation [9]

$$S(\omega) = \int_{-\infty}^{\infty} s(t)\, e^{-i\omega t}\, dt \tag{3.12}$$

The Fourier transform of a real signal is always Hermitian symmetrical, meaning that

$$S(\omega) = S^*(-\omega)$$

in which the asterisk stands as complex conjugate; often $S(\omega)$ is called as the *spectrum* of $s(t)$, but in fact such designation should be reserved for $|S(\omega)|$, which is also called the *amplitude spectral density*. Given $S(\omega)$ it is possible to recover $s(t)$ using the *inverse Fourier transform*, given by

$$s(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} S(\omega)\, e^{i\omega t}\, d\omega \tag{3.13}$$

This expression is also known as *Fourier synthesis*. In signal processing the quantity $|S(\omega)|^2$ is called the *power spectral density* (abbreviated as PSD). In particular *Parseval's theorem* states that the energy of a signal is the same in the frequency and in the time domain:

$$\int_{-\infty}^{\infty} |s(t)|^2\ dt = \int_{-\infty}^{\infty} |S(\omega)|^2\ d\omega \tag{3.14}$$

In the remaining of this book, to simplify matters, we will use the notation

$$S(\omega) = \mathrm{FT}\,[s(t)] \qquad \text{and} \qquad s(t) = \mathrm{FT}^{-1}\,[S(\omega)]$$

Let us notice, in particular, that

$$\mathrm{FT}^{-1}\left[S(\omega)e^{-i\omega\tau}\right] = s(t-\tau)$$

and that

$$\mathrm{FT}^{-1}\left[S^*(\omega)\right] = s(-t)$$

- The convergence of the integral given by Eq. (3.12) is guaranteed for a vast class of functions $s(t)$, periodic or non-periodic, except in the case of a stochastic process, which compromises the existence of the corresponding $S(\omega)$. This important restriction seems to compromise the calculation of the PSD of a random signal; however, as will be clarified in the discussion that follows, the discretization of the signal can be used to solve this issue in a satisfactory manner.
- The discrete equivalents of the Fourier transform and of the inverse Fourier transform correspond to the *Discrete Fourier Transform* and to the *Inverse Discrete Fourier Transform*, defined as [9]

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi kn/N} \tag{3.15}$$

where $k = 0, 1, 2, \ldots N-1$, and

$$x_n = \frac{1}{N}\sum_{k=0}^{N-1} X_k e^{i2\pi kn/N} \tag{3.16}$$

where $n = 0, 1, 2, \ldots N-1$. Parseval's theorem for the discrete case can then be written as [10]

$$N\Delta t \sum_{n=0}^{N-1} |x_n|^2 = \frac{1}{\Delta t}\sum_{k=0}^{N-1} |X_k|^2 \tag{3.17}$$

where Δt stands for the sampling interval. The estimation of the PSD of a discrete signal, even if contaminated with noise, can be achieved using a signal *correlogram* or, alternatively, using a signal *periodogram*, which are discussed in detail in the literature of signal processing (see, for instance, [10, 11]). On the other side, the generation of noise with a particular colour should be based on the definition of a filter with the desired noise properties and the corresponding filtering of white noise, which will be exemplified in Sect. 3.3.8 for the case of pink noise. The differentiation between white noise and noise of a particular colour can be achieved without using a frequency spectrum, through the calculation of the signal *autocorrelation*, which will be discussed in Sect. 3.3.9.

- The *spectrogram* is a tool of signal processing, which allows to represent a signal simultaneously in the time and in the frequency domains. To obtain such representation the signal is divided into intervals with a duration ΔT, and for each interval it is calculated a Fourier transform, which is representative of the temporal centre of the given interval. The frequency resolution Δf of this representation is limited by [12]:

$$\Delta f \Delta T = 1 \tag{3.18}$$

  Therefore, if the interval is too small the frequencies within the interval will not be properly represented; if the interval is too large then temporal information will be lost relative to the behaviour of the signal in the time domain.

Having clarified these issues we can now proceed with the discussion of signals, often referred to in the literature of Underwater Acoustics.

3.3.1 The Unit Impulse

The *unit impulse* $\delta(t)$ (or Dirac's "delta" function) is an ideal signal mathematically defined through the relationship [9]

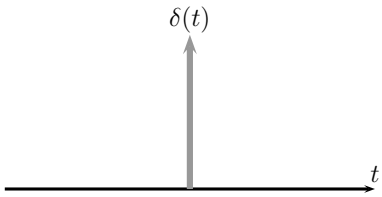


Fig. 3.4 Representation of a unit impulse

$$\int_{-\infty}^{\infty} s(t)\ \delta(t-\tau)\ dt = s(\tau) \tag{3.19}$$

with the properties

$$\delta(t) = 0 \quad \text{when } t \neq 0 \quad \text{and} \quad \int_{-\infty}^{\infty} \delta(t)\ dt = 1$$

(see Fig. 3.4). In particular it is easy to prove that FT $[\delta(t)] = 1$, meaning that a unit impulse in the time domain corresponds to a constant spectrum in the frequency domain.

3.3.2 The Rectangular Signal

The *rectangular* signal $\Pi(t)$ is another ideal signal, mathematically defined as [9]

$$\Pi(t) = \begin{cases} 0 & \text{when } |t| > 1/2 \\ 1/2 & \text{when } |t| = 1/2 \\ 1 & \text{when } |t| < 1/2 \end{cases} \tag{3.20}$$

In particular it can be proved that

$$\text{FT}\,[\Pi(t)] = \text{sinc}\left(\frac{\omega}{2\pi}\right)$$

where the "sinc" function is defined as

$$\text{sinc}(x) = \frac{\sin(\pi x)}{\pi x}$$

As shown in Fig. 3.5 the sinc function is an even function, with a maximum at zero, an oscillatory behaviour and an amplitude which quickly decreases. A sinc signal in the time domain would exhibit a rectangular spectrum in the frequency domain.

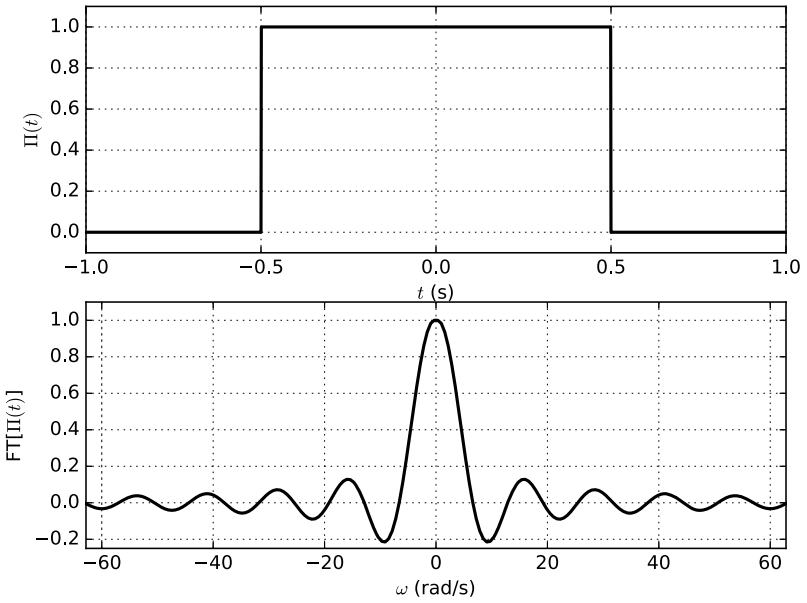


Fig. 3.5 Signal $\Pi(t)$ in the time domain and corresponding Fourier transform

The rectangular signal can be "squeezed" or "widened" dividing t by a width T; in such case one would have that

$$\mathrm{FT}\left[\Pi\left(\frac{t}{T}\right)\right] = T\ \mathrm{sinc}\left(\frac{\omega T}{2\pi}\right)$$

Ignoring the amplitude factor one can conclude that a "thin" rectangular signal has a "broad" sinc in the frequency domain, and vice versa.

3.3.3 Tones

A *tone* (also known as a *Continuous Wave* signal, or simply, a *CW* signal) is just a temporal sinusoid with a single frequency f:

$$s(t) = \cos(\omega t)$$

where

$$\omega = 2\pi f$$

The choice of a cosine function is arbitrary; it can be replaced with a sine function if the condition $s(0) = 0$ is required; a tone of a very short duration is usually called a

burst. In particular, the relation between two frequencies is called an *acoustic interval* [12]

$$i_a = f_1/f_2$$

Such interval is called an *octave* when $f_1 = 2f_2$.

For obvious reasons a *multitone* is a sum of several tones:

$$s(t) = \sum_{i=1}^{N} a_i \cos(\omega_i t)$$

where

$$\omega_i = 2\pi f_i$$

It is a common practice to add a random phase to each component of a multitone to avoid the eventual superposition of maxima and minima of different tones; in this way, it is possible to distribute the energy of the signal in a more uniform manner. Many of the *pings* used in submarine detection corresponded originally to multitones with logarithmically decreasing amplitudes; the main goal was to generate a signal with an echo that could be clearly distinguished from any environmental sounds.

The generation of a multitone with three tones is exemplified in the Python Code Snippet 3.5, which generates Fig. 3.6; the code also includes a preliminary demonstration of *signal filtering*. Figure 3.6a suggests that $s(t)$ is the result of the superposition of periodical signals; the PSD shown in Fig. 3.6b allows to infer that the multitone is composed of three tones, with similar amplitudes; the spectrogram of Fig. 3.6c reveals that the emission of each tone takes place without interruptions along the entire duration of the signal. Figure 3.6d shows the satisfactory result of filtering, which consisted simply in "slicing" $S(\omega)$ with rectangular functions in the frequency domain, preserving only the component of lowest frequency. In practice signal filtering is rarely accomplished with rectangular functions in the frequency domain, because a side effect of their usage is often reflected in the filtered signal through the oscillatory behaviour of the corresponding $\text{sinc}(t)$ functions. Signal filtering will be discussed in more detail in Sect. 3.3.8.

Code Snippet 3.5 Generation of a multitone and filtering using rectangular windows.

```
from scipy import signal
from numpy import *
from matplotlib.pyplot import *

f1 = 5
f2 = 50
f3 = 75
sampling = 500
nyquist = sampling/2
dt = 1.0/sampling
```

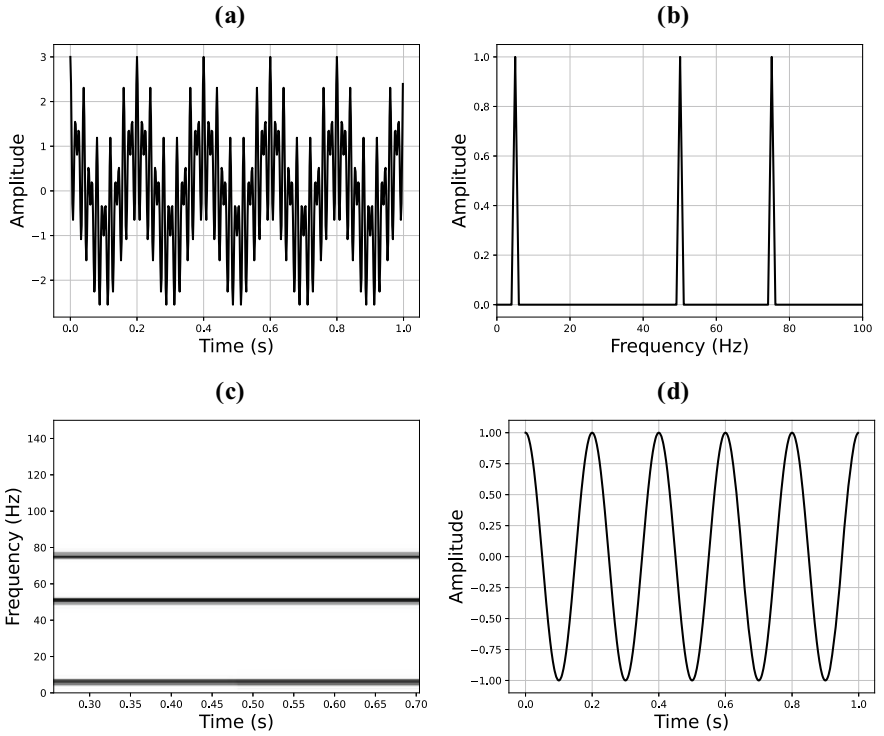


Fig. 3.6 Multitone: **(a)** in the time domain; **(b)** in the frequency domain; **(c)** spectrogram; **(d)** filtered signal, recovering the tone of lowest frequency

```
t = arange(0,1,dt)
w1 = 2*pi*f1
w2 = 2*pi*f2
w3 = 2*pi*f3
s = cos( w1*t ) + cos( w2*t ) + cos( w3*t )
S = fft.fft(s)
Saux = fft.fft(s)

P = real( S*conj( S ) )
P = P/max( P )
frequencies = linspace(0,sampling,sampling)
# Improvised filter,
# remove undesired components:
f = abs( frequencies - f1 )
dummy,i1 = f.min(0),f.argmin(0)
f = abs( frequencies - sampling + f1 )
dummy,i2 = f.min(0),f.argmin(0)
Saux[i1+10:i2-10] = 0;
sf = real( fft.ifft(Saux) )

figure(1)
```

```
plot(t,s,'k',linewidth=2)
grid(True)
xlabel('Time (s)',fontsize=18)
ylabel('Amplitude',fontsize=18)

figure(2)
plot(frequencies,P,'k',linewidth=2)
xlim(0,f3+25)
grid(True)
xlabel('Frequency (Hz)',fontsize=18)
ylabel('Amplitude',fontsize=18)

ef,et,Sxx = signal.spectrogram(s,sampling)

figure(3)
imshow(Sxx, cmap='Greys',
extent=[et[0],et[-1],ef[0],ef[-1]],
aspect='auto',origin='lower')
xlabel('Time (s)',fontsize=18)
ylabel('Frequency (Hz)',fontsize=18)
ylim(0,150)

figure(4)
plot(t,sf,'k',linewidth=2)
xlabel('Time (s)',fontsize=18)
ylabel('Amplitude',fontsize=18)
grid(True)
show()
```

If the number of superimposed tones is high enough then the multitone becomes a *pulse*, that is, it becomes a signal with a large frequency spectrum and a short duration (see Fig. 3.7 and Python Code Snippet 3.6).

Code Snippet 3.6 Generation of a pulse through tone superposition.

```
from scipy import signal
from numpy import *
from matplotlib.pyplot import *

nfreqs = 101
fmin = 20
fmax = 100
f = linspace(fmin,fmax,nfreqs)
sampling = 1000
nyquist = sampling/2
dt = 1.0/sampling
t = arange(0,1,dt)
w = 2*pi*f
s = zeros(sampling)
for i in range(nfreqs):
    s = s + cos( w[i]*(t-0.5) )

s = s/max( abs( s ) )
```

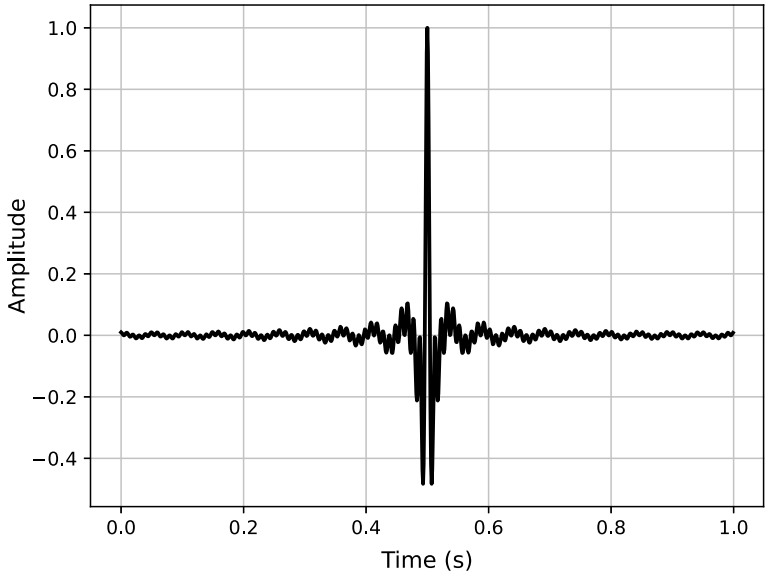


Fig. 3.7 Generation of a pulse through tone superposition

```
figure(1)
plot(t,s,'k',linewidth=2)
grid(True)
xlabel('Time (s)',fontsize=18)
ylabel('Amplitude',fontsize=18)

ef,et,Sxx = signal.spectrogram(s,
sampling,noverlap=250)

figure(3)
pcolormesh(et,ef,Sxx,cmap='Greys',shading='auto')
xlabel('Time (s)',fontsize=18)
ylabel('Frequency (Hz)',fontsize=18)
ylim(0,150)

show()
```

3.3.4 Frequency Modulation

Signals with a frequency modulation are often called *chirps* and are described by the expression

$$s(t) = \cos[\phi(t)]$$

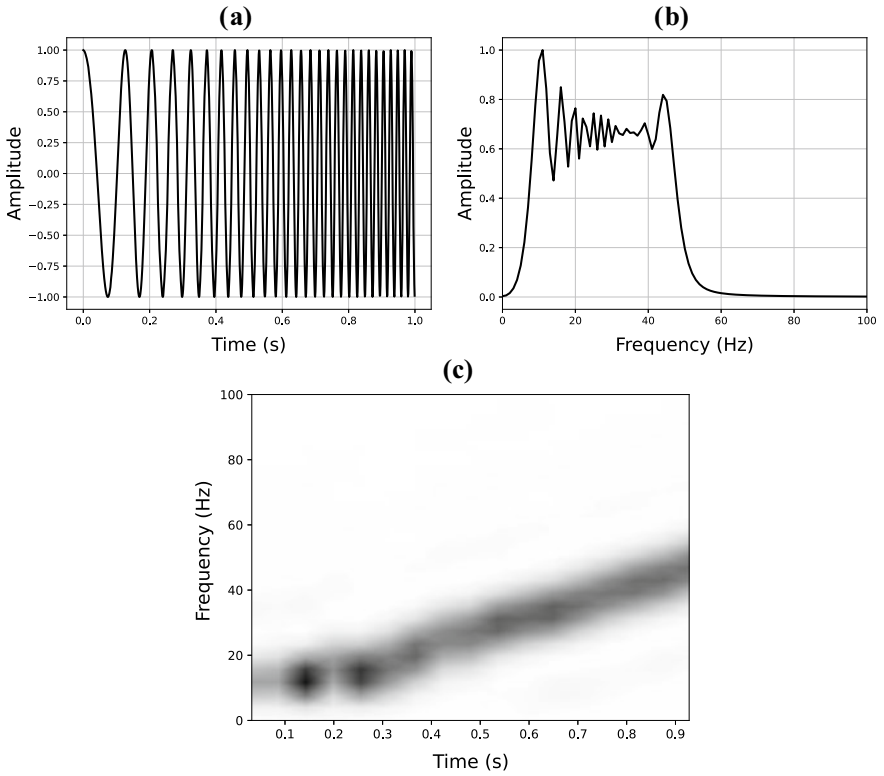


Fig. 3.8 A chirp with a linear modulation of frequency: **(a)** in the time domain; **(b)** in the frequency domain; **(c)** spectrogram

where the signal frequency is given by

$$\omega = \frac{d\phi}{dt}$$

If the frequency increases over time the signal is a called an *upchirp*, otherwise it is called a *downchirp*. In the case of a linear modulation of frequency one can write that

$$\omega = kt + \omega_0 \quad \text{and} \quad \phi(t) = \frac{1}{2}kt^2 + \omega_0 t$$

and the corresponding chirp is called an *lfm* signal (abbreviation of *linear frequency modulation*). Depending on the type of application one might consider other modulations of frequency. The Python Code Snippet 3.7 shows how to generate an lfm signal with an initial frequency, low enough to illustrate the progressive variation of frequency (see Fig. 3.8a). Additionally, Fig. 3.8b, c shows the corresponding Fourier transform and spectrogram.

Code Snippet 3.7 Generation of a chirp with a linear modulation of frequency.

```
from scipy import signal
from numpy import *
from matplotlib.pyplot import *

sampling = 1000
nyquist = sampling/2
dt = 1.0/sampling
t = arange(0,1,dt) # tmax = 1 s
s = zeros( sampling )
f0 = 5.0
f1 = 5.0
f2 = 50.0
k = (f2-f1) # divide by tmax if tmax ~= 1
freq = k*t + f0
w = 2.0*pi*freq
fase = cumsum( w )*dt # Phase = integral of w(t)
s = cos( fase )
S = fft.fft( s )
PSD = real( S*conj( S ) )
PSD = PSD/max( PSD )
frequencies = linspace(0,sampling,sampling)

figure(1)
plot(t,s,'k',linewidth=2)
grid(True)
xlabel('Time (s)',fontsize=18)
ylabel('Amplitude',fontsize=18)

figure(2)
plot(frequencies,PSD,'k',linewidth=2)
xlim(0,2*f2)
grid(True)
xlabel('Frequency (Hz)',fontsize=18)
ylabel('Amplitude',fontsize=18)

ef,et,Sxx = signal.spectrogram(s,
sampling,nperseg=64,nfft=256)

figure(3)
pcolormesh(et,ef,Sxx,cmap='Greys',shading='gouraud')
ylim(0,2*f2)
xlabel('Time (s)',fontsize=18)
ylabel('Frequency (Hz)',fontsize=18)

show()
```

3.3.5 M-sequences

Frequency modulation was addressed in the previous section; however, modulation can also be considered for the phase of the signal, independently of its frequency. In the case of *M-sequences* this modulation can be achieved using the so-called *maximum length sequences* (abbreviated as *MLS*), which constitute a particular type of pseudo-random binary sequences [13]; the usage of an MLS imposes a limit on

the number of "bits" that can be transmitted. Without addressing specific details let us state that an M-sequence can be written as

$$s(t) = \cos(\omega t + m\pi)$$

where m represents a bit that can take the value 0 or 1 every nT seconds, with $T = 2\pi/\omega$; in order for $s(t)$ to represent an M-sequence the sequence of bits $m_1, m_2, m_3, \ldots$ should correspond to an MLS, which can seem random at first sight; however, the MLS sequence has the property of having a very thin autocorrelation, and this property has the advantage of allowing to generate signals, which mitigate the *intersymbolic interference*, abbreviated as *ISI* (autocorrelation will be discussed in detail in Sect. 3.3.9, and ISI will be addressed in Sect. 3.4). For instance, during the global transmissions of the *Heard Island Feasibility Test* (an island located in the south of the Indian Ocean), with receiving stations in the North and South Atlantic, M-sequences with 255, 511, 1023 and 2047 bits were transmitted, with a carrier frequency 57 Hz. The experiment demonstrated that it was possible to coherently process the receiving signals at a global scale, with acceptable signal-to-noise ratios [14].

Phase modulation with an MLS constitutes a particular case of *Binary Phase Shift Keying*, abbreviated as *BPSK*; the modulation is binary because the phase can change between 0 and π radians, which encode the bits 0 and 1. Taking the risk of drifting a bit off-topic it is worth mentioning that phase modulation can be accomplished also with pairs of bits, in what is known as *Phase Shift Keying* (abbreviated as *PSK*). An analogy of PSK in terms of amplitude corresponds to *Amplitude Shift Keying* (abbreviated as *ASK*), which relies on amplitude instead of phase to encode different bits; another analogy can be introduced with *Frequency Shift Keying* (abbreviated as *FSK*). While ASK and FSK are in fact of extreme importance in Underwater Acoustics their discussion is too extensive to be included here. Readers wishing to know more about the subject are invited to consult [11]. The Python Code Snippet 3.8 exemplifies the generation of an M-sequence, in which each bit has a duration of 0,2 s; however, in order to make the code as compact as possible, the random

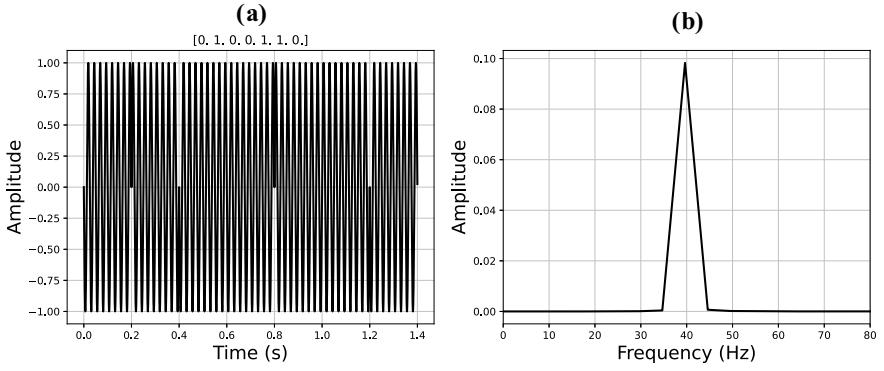


Fig. 3.9 M-sequence: **(a)** in the time domain; **(b)** PSD

sequence of bits is generated in such a way that might (or might not) correspond exactly to an MLS; in this way, the result shown in Fig. 3.9a is one of many possible, and leaves aside the confirmation that the bit sequence is in fact an MLS; Fig. 3.9b shows the corresponding PSD calculated with the method of the periodogram.

Code Snippet 3.8 Generation of an *M*-sequence.

```
from scipy import signal
from numpy import *
from matplotlib.pyplot import *

tmin        = 0.0
tmax        = 1.4
sampling    = 10000
nyquist     = sampling/2
dt          = 1.0/sampling
freq        = 40
w           = 2*pi*freq; T = 1.0/freq
duration_bit    = 0.2
t = arange(0,duration_bit,dt)
nbits       = int( round(tmax/duration_bit) )
sr = random.rand( nbits )
sequence = around( random.rand(nbits) )
thesequence = str( sequence )

um = sin( w*t )
s = []
for i in range( nbits ):
    if sequence[i] == 1:
        s = hstack((s,um))
    else:
        s = hstack((s,-um))

t = arange(0,tmax,dt)
frequencies, PSD = signal.periodogram(s,
sampling,nfft=2018) # periodogram

figure(1)
plot(t,s,'k',linewidth=2)
grid(True)
xlabel("Time (s)",fontsize=18)
ylabel("Amplitude",fontsize=18)
title( thesequence )

figure(2)
plot(frequencies,PSD,'k',linewidth=2)
xlim(0,2*freq)
grid(True)
xlabel("Frequency (Hz)",fontsize=18)
ylabel("Amplitude",fontsize=18)

show()
```

3.3.6 Wavelets

The *wavelets* are a family of functions often defined in the time domain, of frequent application in signal processing, in particular for signal filtering [12]. Haar wavelet, for instance, is defined as

$$\Psi(t) = \begin{cases} 0 & \text{when} \quad 0 \leq t \leq 1/2 \\ -1 & \text{when} \quad 1/2 \leq t \leq 1 \\ 0 & \text{remaining values of } t \end{cases}$$

with a scale function $\phi(t) = 1$ when $0 \leq t \leq 1$ and $\phi(t) = 0$ for other values of t; this wavelet was introduced by the physicist Paul Levy in 1930 to study the Brownian motion since it was easier to manipulate than the corresponding Fourier basis; Haar's wavelets can be simply seen as a positive rectangular signal, followed by a negative rectangular signal. Ricker's wavelet, also known as the "Mexican hat", is defined as [15]

$$\Psi(t) = \left(1 - 2x^2\right) e^{-x^2}$$

where $x = \pi f_{dom} t$; this wavelet can be used as a passband signal filter around a "dominating" wavelet frequency f_{dom}.

The above wavelets, often referred when the topic is introduced, represent a minor fraction of the different types of wavelets in current use; however, a deeper discussion of this subject goes far beyond the objectives of the *fundamentals*. As a concluding remark it is important to notice that, as in the case of the Fourier transform, any wavelet can be used to introduce a time and frequency representation of a signal, with the existence of the inverse transform; however, instead of "blocks" with fixed resolutions Δt and Δf a wavelet representation resorts to "large" blocks for lower frequencies and "small" blocks for higher frequencies, an advantage that can be particularly useful in the identification of *transient* signals, that is, of non-stationary signals.

3.3.7 The Analytic Signal

The spectrogram is certainly a powerful tool of signal processing that allows to identify the temporal distribution of frequencies in a given signal, but it is not the only one. Let us consider a real signal $s(t)$; the *analytic signal* is a complex entity $s_a(t)$ associated with $s(t)$ and defined as [9]

$$s_a(t) = s(t) + i\hat{s}(t) \tag{3.21}$$

where $\hat{s}(t)$ corresponds to the *Hilbert transform* of $s(t)$:

$$\hat{s}(t) = \frac{1}{\pi} \int_{-\infty}^{\infty} \frac{s(\tau)}{t - \tau} \, d\tau \tag{3.22}$$

For instance, the analytic signal of a tone $s(t) = \cos(\omega t)$ is given by

$$s_a(t) = \cos(\omega t) + i \sin(\omega t)$$

Additionally, it is called *instantaneous amplitude* or signal *envelope* the function defined as

$$e(t) = |s_a(t)| \tag{3.23}$$

and the argument of the analytic signal is called the *instantaneous phase*

$$\phi(t) = \arg [s_a(t)] \tag{3.24}$$

The derivative of this phase allows to calculate the signal *instantaneous frequency*:

$$f(t) = \frac{1}{2\pi} \frac{d\phi}{dt} \tag{3.25}$$

From a numerical point of view the phase is to be represented without discontinuities of 2π, this is called phase *unwrapping*. The calculation of the envelope and instantaneous frequency of a signal is illustrated in Fig. 3.10, which is generated by the Python Code Snippet shown in 3.9. In the code the signal $s(t)$ is idealized as a

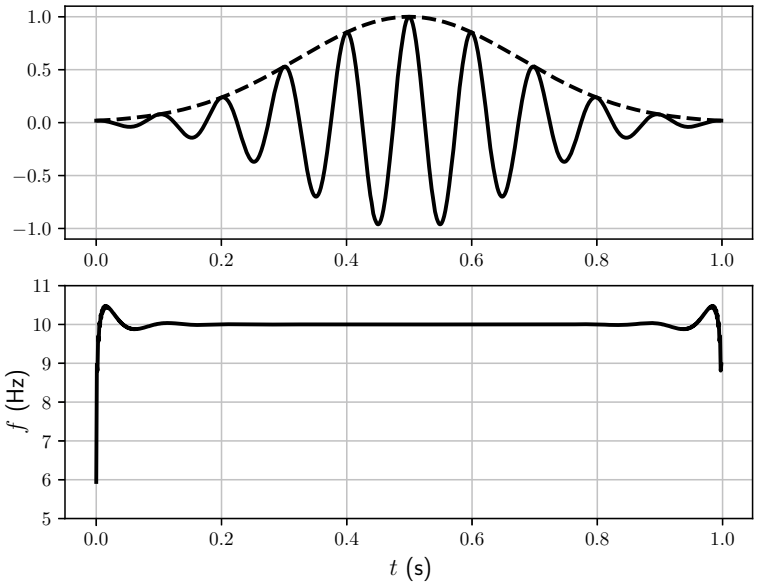


Fig. 3.10 Signal envelope and instantaneous frequency

tone with a frequency 10 Hz, which is modulated by a Gaussian centred at the instant 0.5 s. The signal's envelope, calculated from the analytic signal of $s(t)$, recovers the modulating Gaussian; the instantaneous frequency, calculated from the phase of the analytic signal, is practically constant and equal to the tone's frequency.

Code Snippet 3.9 Signal envelope and instantaneous frequency.

```
from scipy import signal
from numpy import *
from matplotlib.pyplot import *

rcParams.update({"text.usetex": True})

sampling = 1000
nyquist = sampling/2
dt = 1.0/sampling
t = arange(0,1,dt)
x = 4*( t - 0.5 )
freq = 10
w = 2*pi*freq
s = exp( -x*x )*cos( w*t )
analytic_signal = signal.hilbert(s)
envelope = abs( analytic_signal)
phase = angle(analytic_signal)
wi = diff( unwrap(phase) )/diff( t )
fi = wi/( 2*pi )

figure(1)
subplot(211)
plot(t,s,'k',linewidth=2)
plot(t,envelope,'k--',linewidth=2)
ylim(-1.1,1.1)
grid(True)
subplot(212)
plot(t[0:-1],fi,'k',linewidth=2)
xlabel(r'$t$ (s)',fontsize=18)
ylabel(r'$f$ (Hz)',fontsize=18)
ylim(5,11)
grid(True)

show()
```

3.3.8 Signal Filtering

Signal filters are functions of frequency which, when multiplied by the signal spectrum, allow to preserve the content of a given set of frequencies mitigating aliasing as much as possible, something which is extremely difficult to achieve in the general

case with a rectangular function in the frequency domain; in order to describe the different types of filters let us introduce the following notation:

- f_N, the Nyquist frequency;
- $f_1 < f_2$, two arbitrary signal frequencies ($f_2 < f_N$).

With this notation we can classify the different types of filters as [16] follows:

- *Low-pass* filters, which preserve frequencies in the interval $[0,\ f_1]$.
- *High-pass* filters, which preserve frequencies in the interval $[f_2,\ f_N]$.
- *Pass-band* filters, which preserve frequencies in the interval $[f_1,\ f_2]$.
- *Stop-band* filters, which preserve frequencies in the intervals $[0,\ f1]$ and $[f_2,\ f_N]$.

A digital filter can be idealized as a function of a complex variable z, which can be represented as

$$H(z) = \frac{B(z)}{A(z)} \tag{3.26}$$

where $A(z)$ and $B(z)$ are polynomials in z with negative powers:

$$A(z) = a_0 + a_1 z^{-1} + a_2 z^{-2} + \ldots a_n z^{-n}$$

and

$$B(z) = b_0 + b_1 z^{-1} + b_2 z^{-2} + \ldots b_n z^{-n}$$

Whatever might be the order of the polynomials its coefficients can be calculated recursively. In the case when $A(z) = 1$ the filter is known as being of the *FIR* type (abbreviation for *Finite Impulse Response*), otherwise is of the *IIR* type (abbreviation for *Infinite Impulse Response*).

The choice of coefficients for $A(z)$ and $B(z)$ is not unique, different criteria allow to generate different filter "flavours", each with its own advantages and limitations; within such flavours one can mention the Butterworth and Chebyshev filters. The application of a Butterworth filter of order $n = 4$ is exemplified in Fig. 3.11, which is generated by the Python Code Snippet 3.10; the example shows how the filter allows to recover a signal which was originally contaminated with white noise.

Code Snippet 3.10 Signal filtering with a Butterworth filter.

```
from scipy.signal import butter, lfilter
from numpy import *
from matplotlib.pyplot import *

rcParams.update({"text.usetex": True})

a = -1.0
b = 1.0
sampling = 1000
nyquist = sampling/2
```

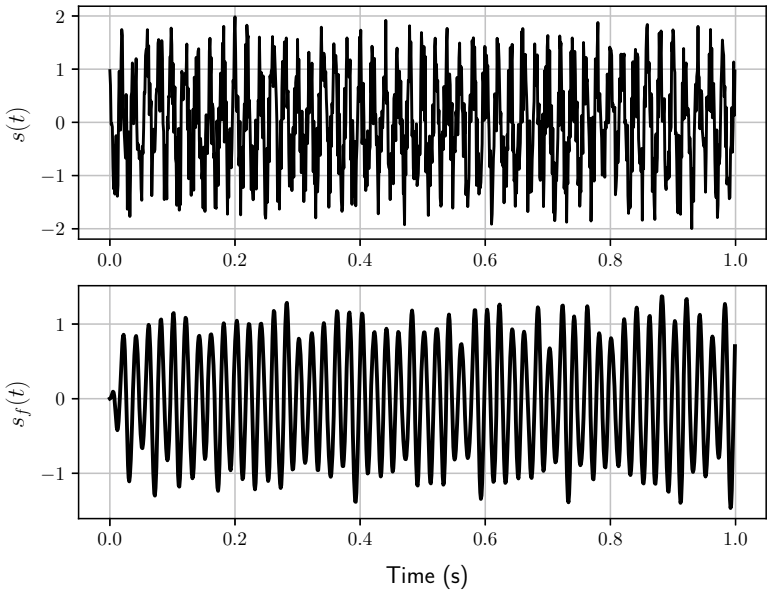


Fig. 3.11 Signal filtering with a Butterworth filter of order $n = 4$

```
n = random.uniform(a,b,sampling) # white noise
dt = 1.0/sampling
t = arange(0,1,dt)
freq = 50
f1 = 20.0
f2 = 80.0
w = 2*pi*freq
s = cos( w*t ) + n # signal and noise
low = f1/nyquist
high = f2/nyquist
# H = B/A:
B, A = butter(4, [low, high], btype='band')
sf = lfilter(B, A, s)

figure(1)
subplot(211), plot(t,s,'k')
ylabel(r'$s(t)$',fontsize=18)
grid(True)
subplot(212), plot(t,sf,'k',linewidth=2)
ylabel(r'$s_f(t)$',fontsize=18)
xlabel('Time (s)' ,fontsize=18)
grid(True)
show()
```

The discussion of how to calculate the filter coefficients goes far beyond the objectives of this book, but we finally have come to the point where the generation of pink noise through signal filtering can be illustrated. In general each colour corresponds

Table 3.1 Parameters used to define filter coefficients according to the noise "colour"

Colour	Power (dB)	Power density (dB)
White	+3	0
Pink	0	−3
Blue	+6	+3
Brown	−3	−6
Violet	+9	+6

to a specific frequency decay, which can be characterized in terms of power and power density per octave (see Table 3.1). For instance, in the case of pink noise such characterization allows to obtain the filter

$$H(z) = \frac{b_0 z - b_1 z^{-1} + b_2 z^{-2} - b_3 z^{-3}}{1 - a_1 z^{-1} + a_2 z^{-2} - a_3 z^{-3}}$$

where

$$\begin{aligned} b_0 &= 0,049922035, \\ b_1 &= 0,095993537, \; a_1 = 2,494956002, \\ b_2 &= 0,050612699, \; a_2 = 2,017265875, \\ b_3 &= 0,004408786, \; a_3 = 0,5221894. \end{aligned}$$

The generation of pink noise with this filter is exemplified in the Python Code Snippet 3.11, which generates the results shown in Fig. 3.12a. As a complement of the result, and to be discussed in detail in Sect. 3.3.9, the code also illustrates in Fig. 3.12b the fundamental differences between white and pink noises through the calculation of the corresponding autocorrelation (normalized) functions; the autocorrelation of white noise exhibits a behaviour similar to an unitary pulse, while the autocorrelation of pink noise has a broader (exponential-like) symmetric decay of amplitude over time.

Code Snippet 3.11 Generation of pink noise with a signal filter.

```
from scipy.signal import correlate, freqz, lfilter
from numpy import *
from matplotlib.pyplot import *

rcParams.update({"text.usetex": True})

sampling = 10000
nyquist = sampling/2
nwhite = random.randn(sampling)
dt = 1.0/sampling
t = arange(0,1,dt)
B = [0.049922035, -0.095993537,
0.050612699, -0.004408786]
A = [1.0 , -2.494956002,
2.017265875, -0.522189400]
w,h = freqz(B,A)
frequencies = w/pi*nyquist
```

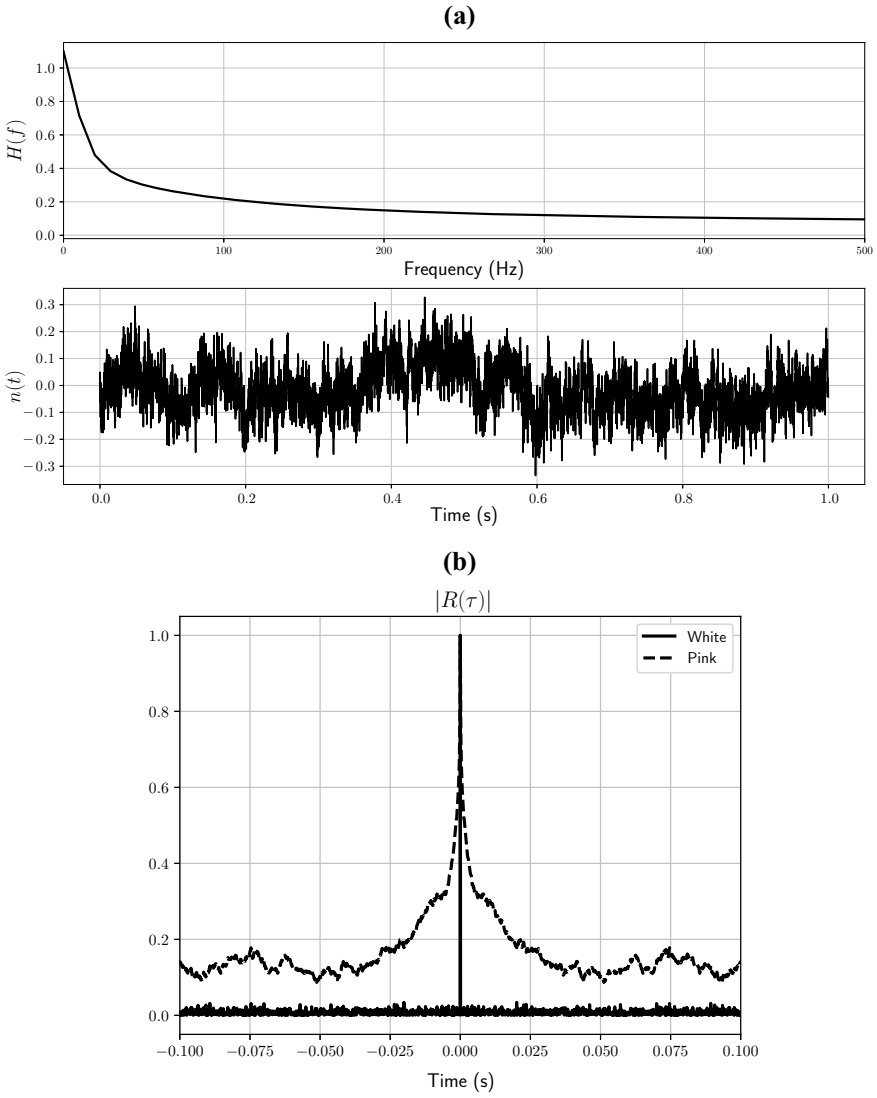


Fig. 3.12 Pink noise generation through white noise filtering: **(a)** the filter that defines the frequency decay (top) and obtained pink noise (bottom); **(b)** normalized autocorrelation function for both white and pink noise

```
npink = lfilter(B, A, nwhite)
Rwhite = correlate(nwhite,nwhite,mode='full')
Rpink = correlate(npink, npink,mode='full')
l = Rwhite.size
tR = linspace(-1.0,1.0,l)
Rwhitenor = abs(Rwhite)/max(abs(Rwhite))
Rpinknor = abs(Rpink)/max(abs(Rpink))
figure(1)
subplot(211)
plot(frequencies,abs(h),'k',linewidth=2)
xlabel('Frequency (Hz)',fontsize=18)
ylabel(r'$H(f)$',fontsize=18)
xlim(0,0.1*nyquist)
grid(True)
subplot(212)
plot(t,npink,'k')
xlabel('Time (s)',fontsize=18)
ylabel(r'$n(t)$',fontsize=18)
grid(True)
figure(2)
plot(tR,Rwhitenor,'k' ,linewidth=2,label='White')
plot(tR,Rpinknor ,'k--',linewidth=2,label='Pink' )
xlabel('Time (s)',fontsize=18)
title(r'$|R(\tau)|$',fontsize=18)
xlim(-0.1,0.1)
legend(loc='best')
grid(True)
show()
```

3.3.9 Signal Correlation

The *correlation* of two signals $s_1(t)$ and $s_2(t)$ is a function which "measures" the similarity of the signals along time, and is defined by the expression [17]

$$C(\tau) = \int_{-\infty}^{\infty} s_1(t)\, s_2(t-\tau)\, dt \tag{3.27}$$

The two signals are said to be *orthogonal* if $C(0) = 0$. The "sliding" nature of the correlation makes it similar to another important relationship, called *convolution*, which combines two functions $s(t)$ and $h(t)$ to generate a third function $p(t)$, given by [9]

$$p(t) = \int_{-\infty}^{\infty} s(\tau)\, h(t-\tau)\, d\tau \tag{3.28}$$

A compact way of writing Eq. (3.28) is to state that

$$p(t) = s(t) \otimes h(t) \tag{3.29}$$

If in Eq. (3.27) we make the choice $s_1(t) = s(t)$, $s_2(t) = h(-t)$ and manipulate the variables in order to replace τ with t, it will follow that $C(t)$ is identical to Eq. (3.28); therefore, one can state that the *convolution* of $s(t)$ with $h(t)$ is equivalent to a *correlation* of $s(t)$ with $h(-t)$.

An important consequence of Eq. (3.28) is the so-called *convolution theorem*; in the case of deterministic signals the convolution of $s(t)$ with $h(t)$ implies that

$$P(\omega) = S(\omega)H(\omega)$$

where

$$S(\omega) = \text{FT}\,[s(t)] \quad , \quad H(\omega) = \text{FT}\,[h(t)] \quad \text{and} \quad P(\omega) = \text{FT}\,[p(t)]$$

in other words, a convolution in the time domain corresponds to a product of the spectra in the frequency domain. In the case of the correlation of two signals the convolution theorem implies that

$$C(\tau) = \text{FT}^{-1}\left[S_1(\omega)S_2^*(\omega)\right]$$

that is, the correlation of two signals in the time domain corresponds to a product of both spectra in the frequency domain, but now with the complex conjugate of the second spectrum. From the point of applications the correlation is often used to detect one signal embedded in another. For obvious reasons the *autocorrelation* function is defined as

$$R(\tau) = \int_{-\infty}^{\infty} s(t)\; s(t-\tau)\; dt = \int_{-\infty}^{\infty} s(t)\; s(t+\tau)\; dt \tag{3.30}$$

The numerical equivalent for a discrete signal with N samples can be written as [9]

$$R(\tau_m) = \frac{1}{N}\sum_{n=0}^{N-|m|-1} s(t_n)s(t_{n+|m|}) \tag{3.31}$$

where the values τ_m are distributed uniformly between $-T$ and T, and T stands for the signal duration. The value of autocorrelation at $\tau = 0$ corresponds to the signal's energy. In terms of convolution Eq. (3.30) implies that

$$R(\tau) = s(\tau) \otimes s(-\tau)$$

Again, according to the convolution theorem, one would have that

$$|S(\omega)|^2 = \text{FT}\,[R(\tau)]$$

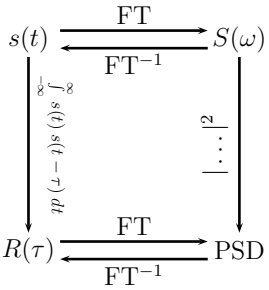


Fig. 3.13 Diagram for time–frequency transformations in the case of deterministic signals

meaning that the signal's PSD corresponds to the Fourier transform of the autocorrelation function. At this stage it is important to remark that while some relationships can take place in two "directions" others are "unidirectional": from the PSD we can calculate the autocorrelation, and vice versa, but from the autocorrelation we cannot calculate the signal and from the PSD we cannot calculate the signal spectrum. The "map" shown in Fig. 3.13 illustrates the time–frequency relationships that exist between the different quantities [18].

In the case of random signals one cannot calculate the signal's Fourier transform, but the autocorrelation is given by

$$R(\tau) = E\left[n(t)n(t-\tau)\right] \tag{3.32}$$

which in fact allows to calculate the signal's PSD through the Fourier transform of the autocorrelation function; the corresponding diagram of time–frequency transformations is shown in Fig. 3.14. In the case of noise sampled at a discrete and finite set of values one cannot calculate the autocorrelation from Eq. (3.32), but the calculation can be based on Eq. (3.31).

In particular, in the case of white noise $n(t)$ with a variance σ^2 it is possible to show that $R(\tau) = \sigma^2\delta(\tau)$ and that the noise PSD is given by σ^2. Therefore, if a signal is contaminated with white noise $r(t) = s(t) + n(t)$, and since the correlation between signal and noise is zero, one can conclude that

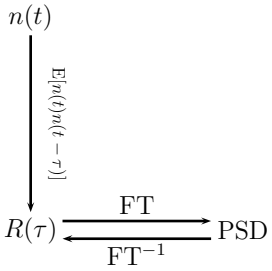


Fig. 3.14 Diagram for time–frequency transformations in the case of random signals

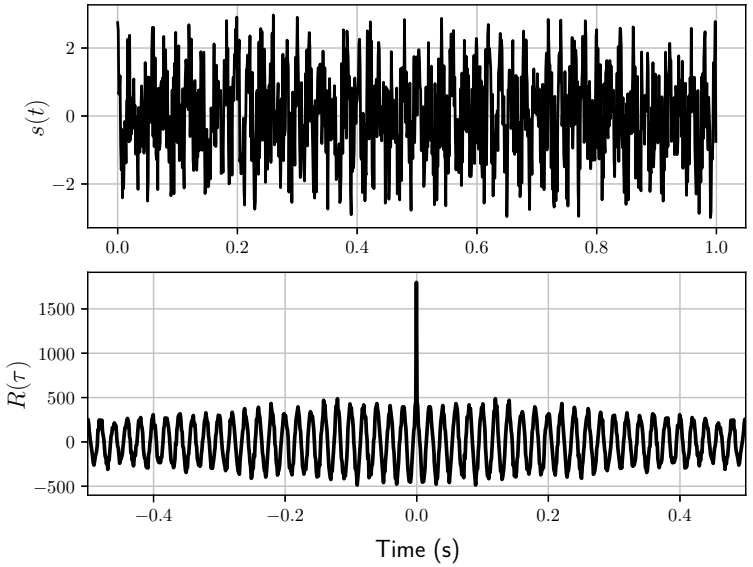


Fig. 3.15 Signal detection: apparent noise (top) and autocorrelation (bottom), revealing the presence of a tone; the intense peak at $\tau = 0$ corresponds to the autocorrelation of white noise

$$R_r(\tau) = R_s(\tau) + \sigma^2 \delta(\tau)$$

which allows the usage of the autocorrelation as a detector of signal in noise. This is exemplified in Fig. 3.15, generated by the Python Code Snippet 3.12, which shows the detection of a tone in a sampled signal, which at first sight seems to contain only noise. Since one is dealing with a signal sampled at discrete intervals the equivalent of Eq. (3.31) is used in the code for the calculation of autocorrelation.

Code Snippet 3.12 Using autocorrelation to detect a signal embedded in noise.

```
from scipy.signal import correlate
from numpy import *
from matplotlib.pyplot import *

rcParams.update({"text.usetex": True})

a = -1.0
b = 1.0
sampling = 1000
n = 2*random.uniform(a,b,sampling) # Noise
dt = 1.0/sampling
t = arange(0,1,dt)
freq = 50
w = 2*pi*freq
s = cos( w*t ) + n # Signal in noise
```

```
# Autocorrelation:
R = correlate( s, s, mode='full')
L = R.size
tR = dt*( arange(L) - L/2 )
figure(1)
subplot(211), plot(t,s,'k')
ylabel(r'$s(t)$', fontsize=18)
grid(True)
subplot(212), plot(tR,R,'k', linewidth=2)
ylabel(r'$R(\tau)$', fontsize=18)
xlabel('Time (s)', fontsize=18)
xlim(-0.5,0.5)
grid(True)
show()
```

3.4 The Impulse Response

In the previous sections we have been ignoring deliberately a fundamental question, namely: who is responsible for the transformation of the *emitted* signal into the *received* signal? The answer is (obviously): the ocean. From the point of view of Physics the ocean corresponds to the *propagation medium*, and the transformation of the signal is defined by a set of differential equations that describe wave propagation, to be discussed in Chaps. 4 and 5. Within the context of signal processing the ocean is generalized as a *system*, which is characterized by a function $h(t)$, and the transformation of the emitted signal $s(t)$ into the received signal $p(t)$ is represented in abstract form as shown in the diagram of Fig. 3.16.

The mathematical relationship between $s(t)$, $h(t)$ and $p(t)$ is a convolution, meaning that $p(t) = h(t) \otimes s(t)$. When we replace $s(t)$ with an unit impulse we can obtain that $p(t) = h(t)$; it is for this reason that $h(t)$ is known as the *impulse response* of the system. In the case of deterministic signals, and according to the convolution theorem, one would have that

$$P(\omega) = H(\omega)S(\omega) \tag{3.33}$$

where

$$P(\omega) = \text{FT}[p(t)] \quad , \quad S(\omega) = \text{FT}[s(t)] \quad \text{and} \quad H(\omega) = \text{FT}[h(t)]$$

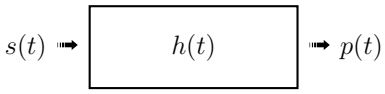


Fig. 3.16 Signal processing: transformation of the signal $s(t)$ into the signal $p(t)$

In order to make clear that we are now in the frequency domain $H(\omega)$ is called the *transfer function*, and its identification as a signal filter should be evident. In such context the received signal can be seen as the result of signal filtering by the system, and this often will manifest itself in two different ways: on one side $h(t)$ *changes* the form of $s(t)$, on another side $h(t)$ *delays* the signal. In a first approximation often used in Underwater Acoustics it is considered that [19]

$$h(t) = \sum_i a_i \delta(t - \tau_i)$$

which implies that

$$p(t) = s(t) \otimes h(t) = \sum_i a_i s(t - \tau_i)$$

in other words, the received signal is nothing more than a sum of delayed and weighted *replicas*; under such approximation the original form is preserved, but this is not the case in real conditions. Even without a change of the signal shape it can happen that two or more replicas interfere with one another, in what is known as *intersymbolic interference* (abbreviated as ISI). The effects of ISI are illustrated in Fig. 3.17, which is generated by the Python Code Snippet 3.13; the code idealizes the transmission of a pair of bits through an "ocean", with the pair of bits corresponding to a Haar wavelet: the positive rectangular signal represents the bit 1 and the negative rectangular signal represents the bit 0. The system impulse response $h(t)$ is idealized as a pair of Gaussians, the second smaller than the first; the received signal is calculated

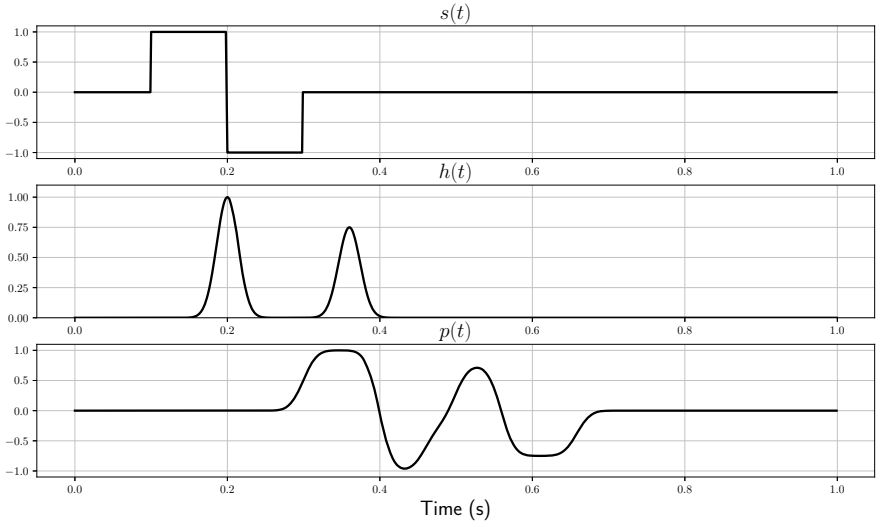


Fig. 3.17 Convolution of the emitted signal with the impulse response: emitted "signal" (top); system impulse response (middle); received signal (bottom), showing ISI

according to Eq. (3.33), and an inverse Fourier transform allows to go back to the time domain. The result shown in Fig. 3.17 indicates that as a result of filtering of the emitted signal with the system impulse response each bit is not only delayed and changed in shape, but the original pair is duplicated leading to ISI in the reception. Without entering specific details it should be remarked that the ISI can be mitigated by filtering the message before transmission with a proper filter, like, for instance, the raised cosine filter [9].

Code Snippet 3.13 Convolution of the emitted signal with the system impulse response.

```
from scipy import *
from numpy import *
from matplotlib.pyplot import *

rcParams.update({"text.usetex": True})

sampling = 1000
dt = 1.0/sampling
t = arange(0,1,dt)

s = zeros(sampling)
s[100:199] = 1
s[200:299] = -1
h = exp( -((t-0.2)/0.02)**2 ) + \
0.75*exp( -((t-0.36)/0.02)**2 )

S = fft.fft( s )
H = fft.fft( h )
P = H*S
p = real( fft.ifft(P) )
p = p/max( abs( p ) )

figure(1)
subplot(311)
plot(t,s,'k',linewidth=2)
title(r'$s(t)$',fontsize=18)
ylim(-1.1,1.1)
grid(True)
subplot(312)
plot(t,h,'k',linewidth=2)
title(r'$h(t)$',fontsize=18)
ylim(0,1.1)
grid(True)
subplot(313)
plot(t,p,'k',linewidth=2)
title(r'$p(t)$',fontsize=18)
ylim(-1.1,1.1)
grid(True)
xlabel('Time (s)',fontsize=18)

show()
```

3.5 Fourier Synthesis

So far we have been discussing the issues of signal modulation starting always from the time domain and moving on to the frequency domain; however, many applications of Underwater Acoustics require the transition in the opposite direction, that is, one needs to generate the signal in the time domain from its components in the frequency domain. For practical effects such generation should take into account the symmetry of the spectrum in the frequency domain and a proper modulation of the signal's phase, something that is not easy to implement in the general case. The Python Code Snippet 3.14 illustrates Fourier synthesis in the particular case of the pulse of Fig. 3.7; the code proceeds along the following stages:

- $S(\omega)$ is defined initially as an empty complex vector, with the number of components corresponding to the sampling frequency.
- $S(\omega)$ is filled symmetrically depending on the pulse frequencies.
- Once $S(\omega)$ is filled it is multiplied by a phase factor, which corresponds in the time domain to a temporal shift.
- The pulse is obtained applying the inverse Fourier transform:

$$s(t) = \mathrm{FT}^{-1}\left[S(\omega)\right]$$

The code algorithm is valid for a pulse with a duration of 1 s; it is left to the reader to generalize the code for the generation of signals with other durations and other frequency bands.

Code Snippet 3.14 Criation of a pulse through Fourier synthesis.

```
from scipy import signal
from numpy import *
from matplotlib.pyplot import *

fmin = 20
fmax = 100
sampling = 1000
nyquist = int( sampling/2 )
frequencies = arange(0,sampling,1)
w = 2*pi*frequencies
S = zeros(sampling) + 1j*zeros(sampling)
dt = 1.0/sampling
t = arange(0,1,dt)
t0 = 0.5
t0 = dt*int( t0/dt )

fpulse = arange(fmin,fmax+1,1)
L = fpulse.size

for i in range(L):
    fi = int( fpulse[i] )
    S[fi]          = ( 1.0 + 1j*0 )*nyquist/L
```

```
        S[sampling-fi] = ( 1.0 + 1j*0 )*nyquist/L

S = S*exp( -1j*w*t0 )
s = real( fft.ifft( S ) )

figure(1)
plot(frequencies, abs(S),'k',linewidth=2)
xlabel('Frequency (Hz)',fontsize=18)
ylabel('Amplitude' ,fontsize=18)
ylim(0,1.1)
grid(True)

figure(2)
plot(t,s,'k',linewidth=2)
xlabel('Time (s)',fontsize=18)
ylabel('Amplitude',fontsize=18)
grid(True)

show()
```

3.6 The Signal-to-Noise Ratio

3.6.1 Definition

The *signal-to-noise ratio* (abbreviated as *SNR*) is defined as the ratio between the signal power and the noise power. In the case of a signal sampled at discrete intervals the SNR can be calculated through the expression

$$\text{SNR} = \frac{\sum_{i=1}^{N} s_i^2}{\sigma^2} \tag{3.34}$$

where σ^2 represents the noise variance; in the general case it is desired to *increase* the SNR.

According to Eq. (3.34) the SNR has no units, something that sometimes can be an inconvenience because small changes in the noise power, for a fixed signal power, can change substantially the value of the SNR; to overcome this problem what in practice is called SNR is the expression

$$\text{SNR} = 10 \log \left(\frac{\text{Signal power}}{\text{Noise power}} \right) \tag{3.35}$$

where log stands for a logarithm in base 10; thus, the definition given by Eq. (3.35) quantifies the SNR in *decibels* (dB). It should be remarked that there is an equivalent

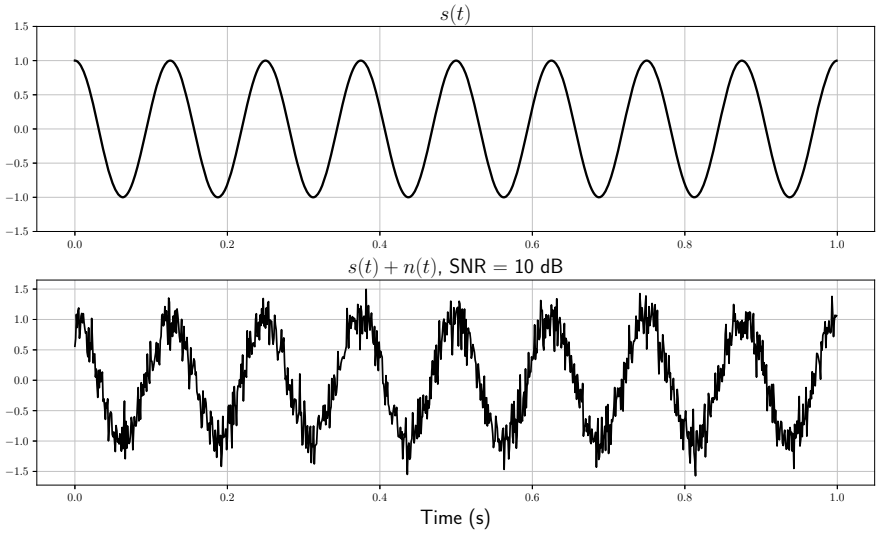


Fig. 3.18 Contamination of a tone with white noise for an SNR of 10 dB

unit based on the logarithm of base e, called *neper* (Np); the conversion factors between dB and Np correspond to

$$1 \text{ Np} = 20 \log e \text{ dB} \qquad \text{and} \qquad 1 \text{ dB} = (1/20) \ln 10 \text{ Np}.$$

3.6.2 Signal Contamination with Noise

Let us consider now the problem of contaminating a signal $s(t)$ with noise $n(t)$ for a given value of the SNR_{dB}. In the general case we have that

$$\text{SNR} = \frac{P_s}{\sigma^2}$$

where P_s represents the signal's power. Solving in order to P_s one can conclude that the desired SNR_{dB} can be obtained as long as the following condition is fulfilled:

$$\sigma = \sqrt{\frac{P_s}{\text{SNR}}}$$

where $\text{SNR} = 10^{\text{SNR}_{dB}}$. In other words, in order to contaminate the signal with noise for a given SNR_{dB} one needs to generate noise scaling its variance. The Python Code Snippet 3.15 exemplifies the contamination of a tone with white noise normally distributed, for a signal-to-noise ratio of 10 dB (see Fig. 3.18).

Code Snippet 3.15 Contaminating a signal with noise.

```
from numpy import *
from matplotlib.pyplot import *

rcParams.update({"text.usetex": True})

SNRdB = 10
SNR   = 10**( SNRdB/10.0 )

thetitle = '$s(t) + n(t)$, SNR = '
thetitle = thetitle + str(SNRdB) + ' dB'
freq     =     8 # Signal frequency
sampling = 1000 # Sampling frequency
nyquist     = sampling/2 # Nyquist frequency
frequencies = linspace(0,sampling,sampling)
dt = 1.0/sampling
t = arange(0,1,dt)
w = 2*pi*freq
s = cos( w*t ) # Tone

Ps = s.var( )

sigma = sqrt( Ps/SNR )
# Signal contamination:
r = s + random.normal(0,sigma,sampling)

figure(1)
subplot(211)
plot(t,s,'k',linewidth=2)
ylim(-1.5,1.5)
grid(True)
title(r'$s(t)$',fontsize=18)
subplot(212)
plot(t,r,'k')
#ylim(-1.5,1.5)
grid(True)
title(thetitle,fontsize=18)
xlabel('Time (s)',fontsize=18)

show()
```

3.6.3 Estimating the SNR

In this section we would like to consider the opposite problem of contamination, that is, to determine the SNR of a given signal or of a set of signals. In the case of a single signal there is the possibility of filtering the signal into two components, the first for the band of interest and the second outside that band. The first component, although contaminated with noise, can be used to estimate the signal's power; the second

component can be used to estimate the noise power. In the case of N synchronized observations of the contaminated signal, and as long as noise is not correlated, the SNR can be increased estimating the original signal as the mean of observations. The signal power can then be estimated from this mean; a sample of "pure" noise can be obtained calculating the difference between the signal mean and one of the N observations, allowing to calculate the noise power. The Python Code Snippet 3.16 exemplifies the estimation of the SNR in such conditions.

Code Snippet 3.16 Estimating the SNR.

```
from numpy import *

freq       =    8  # Signal frequency
sampling   =  500  # Sampling frequency
nyquist    =  sampling/2  # Nyquist frequency
dt  =  1.0/sampling
t  =  arange(0,1,dt)
w  =  2*pi*freq
s  =  cos( w*t )  # Tone

# Estimate the SNR wih a set
# of contaminated signals:

nsignals  =  101
r  =  zeros((nsignals,sampling))
for i in range(nsignals):
    r[i,:]  =  s  +  random.randn( t.size )

se  =  mean( r, axis=0 )  # Estimated signal
ne  =  r[-1,]  -  se      # Estimated noise
Ps  =  se.var()
Pn  =  ne.var()
SNR    =  abs( Ps/Pn )
SNRdB  =  10*log10( SNR )
thesnr  =  "Estimated SNR  =  "  +  str(SNRdB)  +  "  dB"
print( thesnr )
```

3.7 Detection Theory

Let us approach the conclusion of this chapter introducing a few concepts from detection theory; to this end we will adapt the discussion presented in [20]. The *minimal* SNR value that allows the detection of a signal with a probability of 50% is called the *detection threshold* (abbreviated as *DT*); it corresponds to the signal-to-noise ratio required for a probability of detection of 50%.

The DT has a random nature because the signal that we want to detect is contaminated with noise. Let us imagine the detection system shown in Fig. 3.19, which

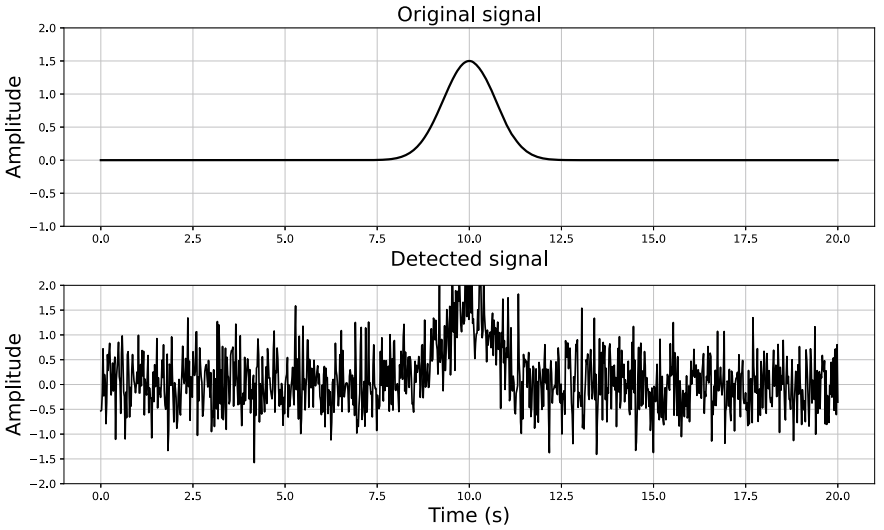


Fig. 3.19 Amplitude of the signal to be detected and of the signal recorded by the detection system

tries to determine if the "echo" of a given signal is present in the signal recorded by the system, where the recording has been contaminated with environmental noise.

Let $A(t)$ be the amplitude of the recorded signal; in order to make a decision the system operator can choose arbitrarily a threshold amplitude A_T, and decide that the signal is present if $A(t) > A_T$; otherwise there is no signal. If the value of A_T is too low the operator will conclude (frequently) that the signal is present, even in situations in which the signal is in fact absent; if the value of A_T is too high the operator will conclude from time to time that the signal is absent, even in situations in which the signal is in fact embedded in noise. Since the noise has a random nature each choice of A_T defines a *probability of detection* $p(D)$, as well as a *probability of false alarm* $p(FA)$. The probability of detection corresponds to the case in which the operator decides (correctly) that the signal to be detected is present in the recorded signal; the probability of false alarm corresponds to the case in which the operator decides (wrongly) that the signal has been detected, when in fact the recording contains only noise. In order to address the calculation of $p(D)$ and of $p(FA)$ let us admit now that the noise has a Gaussian distribution, with mean μ_n and variance σ_n^2. Its PDF can be written as

$$p(A) = \frac{1}{\sqrt{2\pi}\sigma_n} \exp\left[-\frac{1}{2}\left(\frac{A-\mu_n}{\sigma_n}\right)^2\right] \tag{3.36}$$

For each value of A_T we can write now that

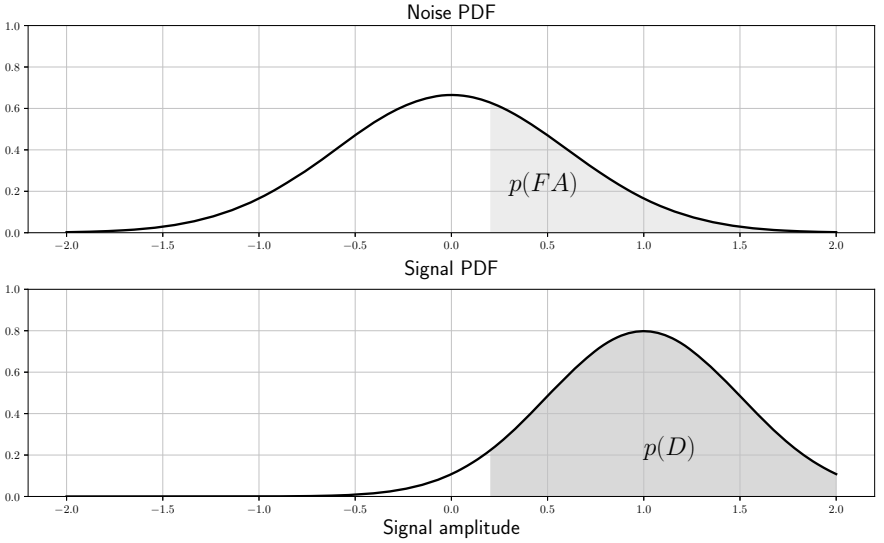


Fig. 3.20 PDFs of the signal and noise, showing the areas that correspond to $p(FA)$ and $p(D)$ for the threshold $A_T = 0, 2$

$$p(FA) = \frac{1}{\sqrt{2\pi}\sigma_n} \int_{A_T}^{\infty} \exp\left[-\frac{1}{2}\left(\frac{A-\mu_n}{\sigma_n}\right)^2\right] dA \tag{3.37}$$

The detected signal, being contaminated with noise, can be characterized also with a mean μ and a variance σ^2; therefore, the probability of detection can be written as

$$p(D) = \frac{1}{\sqrt{2\pi}\sigma} \int_{A_T}^{\infty} \exp\left[-\frac{1}{2}\left(\frac{A-\mu}{\sigma}\right)^2\right] dA \tag{3.38}$$

The two means and variances allow to define the *detection index*, given by

$$d = \frac{2\left(\mu - \mu_n\right)^2}{\left(\sigma^2 + \sigma_n^2\right)} \tag{3.39}$$

The graphic representation of the PDFs for the signal and the noise, and the areas that correspond to $p(FA)$ and $p(D)$ for $A_T = 0, 2$ are shown in Fig. 3.20, which is generated by the Python Code Snippet 3.17.

Code Snippet 3.17 Calculation of the probabilities of false alarm and detection.

```
from numpy import *
from matplotlib.pyplot import *

rcParams.update({"text.usetex": True})

# Noise and signal means and variances:
mun    = 0.0
sigman = 0.6
mus    = 1.0
sigmas = 0.5
xt = 0.2

n = 1000

x = linspace(-2,2,n)
y = (x-mun)/sigman
pdfn = 1/(sqrt(2*pi)*sigman)*exp(-0.5*y*y)
y = (x-mus)/sigmas
pdfs = 1/(sqrt(2*pi)*sigmas)*exp(-0.5*y*y)
yaux = zeros(n)

figure(1)
subplot(211),plot(x,pdfn,'k',linewidth=2)
fill_between(x,pdfn,yaux,where=x>xt,
facecolor='0.9')
text(0.3, 0.2, r'$p(FA)$', fontsize=22)
title('Noise PDF',fontsize=18)
ylim(0,1)
grid(True)
subplot(212),plot(x,pdfs,'k',linewidth=2)
fill_between(x,pdfs,yaux,where=x>xt,
facecolor='0.8')
text(1.0, 0.2, r'$p(D)$', fontsize=22)
title('Signal PDF',fontsize=18)
ylim(0,1)
grid(True)
xlabel('Signal amplitude',fontsize=18)

show()
```

The choice of "ideal" values of $p(D)$ and $p(FA)$ can be clarified through the calculation of the *Receiver Operating Characteristic* curve (abbreviated as *ROC* curve), which shows $p(D)$ as a function of $p(FA)$ (see Fig. 3.21 and Python Code Snippet 3.18); the greater the area under the curve the better the performance of the detection system. The behaviour of each ROC curve depends on the specific value of the detection index, which imposes a restriction on the minimal tolerance to the probability of false alarm for a given value of the probability of detection, which in most practical cases can correspond to 50%. For instance, in the case of Fig. 3.21, a probability of detection of 85% implies an operational tolerance to a probability of false alarm of

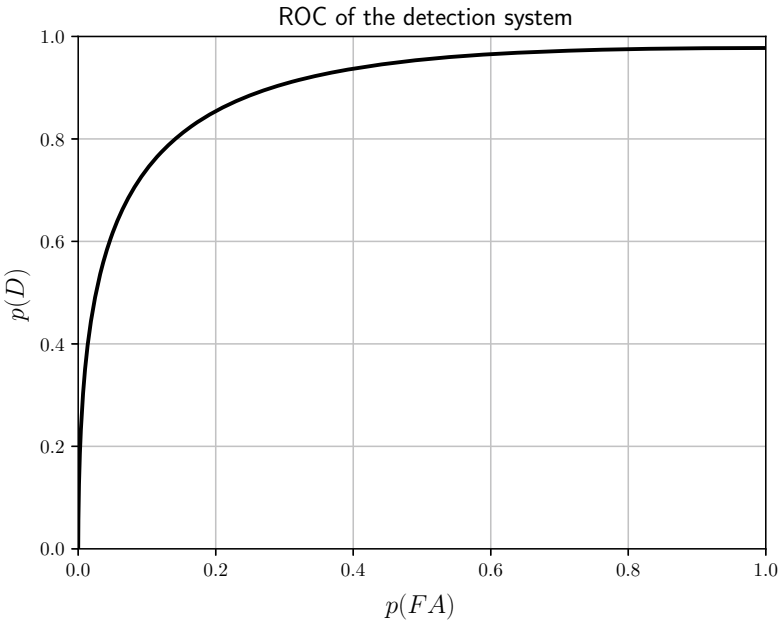


Fig. 3.21 The ROC curve for the signal and noise PDFs

20%, while a probability of detection of 95% implies a tolerance to a probability of false alarm even higher, of 40%.

Code Snippet 3.18 Calculation of a ROC curve.

```
from numpy import *
from matplotlib.pyplot import *

rcParams.update({"text.usetex": True})

# Signal and noise means and variances:
mun    = 0.0
sigman = 0.5
mus    = 1.0
sigmas = 0.5

n = 1000

# Detection index:
a = mus - mun
b = sigmas**2 + sigman**2
d = a**2/( 0.5*b )
print(d)

x = linspace(-2,2,n)
y = (x-mun)/sigman
```

```
pdfn = 1/( sqrt(2*pi)*sigman )*exp( -0.5*y*y )
y = (x-mus)/sigmas
pdfs = 1/( sqrt(2*pi)*sigmas )*exp( -0.5*y*y )

dx = x[1]-x[0]
pFA = zeros( n )
pD = zeros( n )

for i in range(n):
    pFA[i] = sum( pdfn[i:n] )
    pD[ i] = sum( pdfs[i:n] )

pFA = dx*pFA
pD = dx*pD

figure(1)
plot(pFA,pD,'k',linewidth=2)
title('ROC of the detection system',fontsize=18)
xlabel(r'$p(FA)$',fontsize=18)
ylabel(r'$p(D)$',fontsize=18)
xlim(0,1)
ylim(0,1)
grid(True)

show()
```

3.8 Cepstrum Analysis

Cepstrum analysis is an area of signal processing, centred in the concept of *cepstrum*[3] [21], this denomination goes hand in hand with a glossary of terms that can easily stand as misspelling errors, such as

- *quefrency* instead of frequency,
- *rahmonics* instead of harmonics,
- *gamnitude* instead of magnitude,
- *dedumolation* instead of demodulation, or
- *liftering* instead of filtering.

In order to understand the origin of these glossary one needs to introduce the fundamental idea of cepstrum analysis. Let us consider a signal and its echo:

$$p(t) = s(t) + \alpha s(t - \tau)$$

it is easy to prove that the PSD of $p(t)$ corresponds to

$$|P(\omega)|^2 = |S(\omega)|^2 \left[1 + \alpha^2 + 2\alpha \cos(\omega\tau)\right]$$

[3] It is supposed to be pronounced as "quepstrum", because "spectrum" and "spektrum" are pronounced in the same way; for obvious reasons, it is often pronounced as "sepstrum".

where $S(\omega) = \mathrm{FT}[s(t)]$ and $P(\omega) = \mathrm{FT}[p(t)]$. The *cepstrum* (*spectrum*, rewritten by switching some letters) of $p(t)$ is defined as

$$C(\omega) = \log|P(\omega)|^2 = \log|S(\omega)|^2 + \log\left[1 + \alpha^2 + 2\alpha\cos(\omega\tau)\right] \tag{3.40}$$

where log can be a logarithm in any basis b. Equation (3.40) shows that $C(\omega)$ can be idealized as a periodic additive function, with a "fundamental frequency" that corresponds to the delay τ of the echo. In conventional signal processing such periodic components can be identified as intense peaks in the signal spectrum. Therefore, the "spectrum" of the cepstrum can be expected to exhibit a spectral peak if the signal contains an echo. This new spectral representation is not a representation in the time domain or in the frequency domain, thus the choice of the denomination *quefrency* (and associated glossary) to refer to the specificities of cepstrum analysis. The magnitude $C(\omega)$ defined by Eq. (3.40) is often referred in the literature as the *power cepstrum*; later discussion lead to the definition of the *complex* and *real* cepstrums of the signal, defined, respectively, as

$$C_c = \mathrm{FT}^{-1}\left[\log P(\omega)\right] \tag{3.41}$$

$$C_r = \mathrm{Re}\left\{\mathrm{FT}^{-1}\left[\log|P(\omega)|\right]\right\} \tag{3.42}$$

In particular the complex cepstrum is reversible, meaning that

$$p(t) = \mathrm{FT}^{-1}\left\{b^{\mathrm{TF}[C_c]}\right\}$$

Figure 3.22, generated by the Python Code Snippet 3.19, illustrates the application of the complex cepstrum to the detection of an echo in a signal $p(t)$, which is constituted by a tone $s(t) = \cos(\omega t)$ to which an echo of smaller amplitude is added, with a delay τ. This case is of interest because it is rather unlikely that the convolution between $p(t)$ and $s(t)$ can be used to determine τ without ambiguity. The cepstrum exhibits several peaks along the quefrency axis; however, the value of the delay becomes evident due to the position of the strongest peak, which is located at the quefrency of 0.19 s.

Code Snippet 3.19 Signal with an echo and corresponding cepstrum.

```
from numpy import *
from matplotlib.pyplot import *

rcParams.update({"text.usetex": True})

sampling = 1000
nyquist = sampling/2
dt = 1.0/sampling
t = arange(0,1,dt)
```

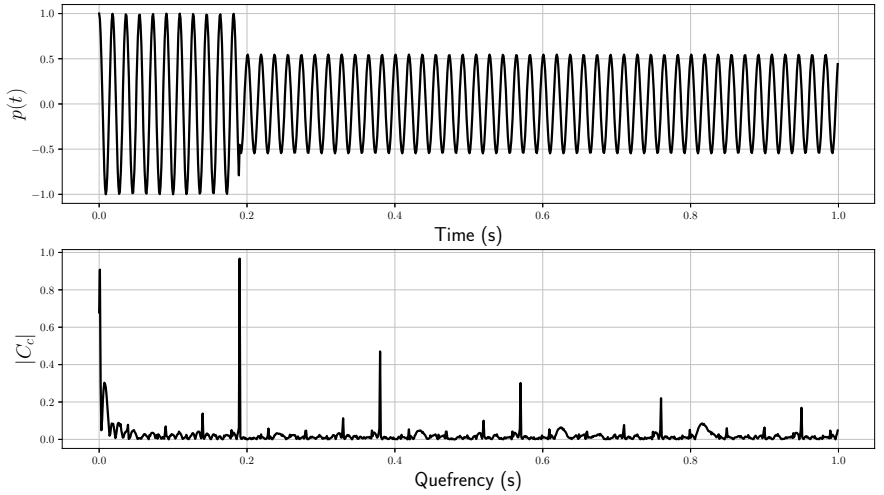


Fig. 3.22 Signal in the time domain, and absolute value of the complex cepstrum in the quefrency domain, with the strongest harmonic at 0,19 s

```
freq = 55
w = 2*pi*freq
s = cos(w*t)
stau = zeros( sampling )
tau = 0.19
a = 0.5
n = int( fix( tau/dt) )
stau[n:] = s[0:sampling-n] # delayed signal
p = s + a*stau # signal plus echo
P = fft.fft( p )
Cc = fft.ifft( log( P ) ) # Complex cepstrum
#Cr = fft.ifft( log( abs( P ) ) ) # Real cepstrum

figure(1)
subplot(211)
plot(t,p,'k',linewidth=2)
xlabel('Time (s)',fontsize=18)
ylabel(r'$p(t)$' ,fontsize=18)
grid(True)
subplot(212)
plot(t,abs(Cc),'k',linewidth=2)
xlabel('Quefrency (s)',fontsize=18)
ylabel(r'$|C_c|$' ,fontsize=18)
grid(True)
#subplot(212)
#plot(t,real(Cr),'k',linewidth=2)
#xlabel('Quefrency (s)')
#ylabel(r'Re$(C_r)$')
#grid(True)
show()
```

Chapter 4
Waves in Boundless Media

4.1 Introduction

We are now going to discuss the problem of signal transmission as a problem of wave propagation. This discussion will proceed in two stages: the first one, presented in this chapter, will consider wave propagation inside a boundless fluid, for a constant speed of propagation. The second one, to be presented in Chap. 5, will address the problem of propagation in continuous media (i.e. both solids and fluids) accounting for the existence of boundaries, and with a variable speed of propagation; such perspective of the problem will allow to introduce the concepts of wave refraction and wave reflection.

4.2 The Wave Equation

Let us consider an acoustic source located at the position given by a vector $\mathbf{r}_s$; when this source transmits a signal $s(t)$ inside a fluid, this signal will propagate from the source to the receiver as a wave. From the point of view of Physics *all* waves should satisfy a differential equation in partial derivatives, called the *wave equation*, which in the case of a pressure wave can be written as [3, 20]

$$\nabla^2 p - \frac{1}{c^2}\frac{\partial^2 p}{\partial t^2} = -s(t)\delta(\mathbf{r} - \mathbf{r}_s) \tag{4.1}$$

where

- $p(\mathbf{r}, t)$ represents the pressure field induced by the propagation of the signal,
- $c = c(\mathbf{r}, t)$ corresponds to the speed of propagation of sound in the fluid,
- ∇^2 is a differential operator in partial derivatives, known as the *Laplacian*.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023

O. C. Rodríguez, *Fundamentals of Underwater Acoustics*,
https://doi.org/10.1007/978-3-031-31319-6_4

The presence of the negative sign in the second term of the equation will be explained in Sect. 5.2.

The wave equation admits analytical solutions in a very limited number of cases. In order to deal with it in a more efficient manner let us apply a Fourier transform to the two sides of the equation, which leads to the *Helmholtz equation*:

$$\nabla^2 P + k^2 P = -S(\omega)\delta(\mathbf{r} - \mathbf{r}_s) \tag{4.2}$$

where

$$k = \frac{\omega}{c} \tag{4.3}$$

is called the *wavenumber*. The mathematical formalism that allows to obtain a general solution of Eq. (4.1) proceeds in the following way:

- Solve the *homogeneous* form of Eq. (4.2):

$$\nabla^2 P_h + k^2 P_h = 0$$

- Use P_h to obtain the transfer function, which in Physics is called the *Green function*:

$$\nabla^2 H + k^2 H = \delta(\mathbf{r} - \mathbf{r}_s)$$

- Represent the solution of Eq. (4.2) as

$$P(\mathbf{r}, \omega) = H(\mathbf{r}, \omega)S(\omega) \tag{4.4}$$

- Use the inverse Fourier transform to obtain the pressure field in the time domain.

Obviously, according to the convolution theorem, we just came back to the relationship

$$p(\mathbf{r}, t) = h(\mathbf{r}, t) \otimes s(t)$$

4.3 Analytical Solutions

Independently of their narrow validity, there are plenty of cases in which the analytical solutions of the wave equation can be of extreme importance. Taking into account that waves are periodic phenomena many of those solutions can be represented as trigonometric functions; however, in order to simplify the manipulation of the expressions we will resort systematically to the use of complex exponentials. The final solution can then be considered as the real (or the imaginary) part of the complex exponential. The sections that follow will present analytical *monochromatic* solutions (i.e. valid only for a single frequency ω), for the homogeneous case of the wave equation in different coordinate systems and for a constant speed of propagation.

4.3.1 Plane Waves

In the case of Cartesian coordinates (x, y, z) one would have that

$$\nabla = \mathbf{e}_x \frac{\partial}{\partial x} + \mathbf{e}_y \frac{\partial}{\partial y} + \mathbf{e}_z \frac{\partial}{\partial z}$$

allowing to write the homogeneous form of the wave equation as

$$\frac{\partial^2 p_h}{\partial x^2} + \frac{\partial^2 p_h}{\partial y^2} + \frac{\partial^2 p_h}{\partial z^2} - \frac{1}{c^2} \frac{\partial^2 p_h}{\partial t^2} = 0$$

The solution is well known and corresponds to

$$p_h(x, y, z, t) = p_0 \exp\left[i \left(\mathbf{k} \cdot \mathbf{r} - \omega t + \phi\right)\right] \tag{4.5}$$

where

$$\begin{aligned} \mathbf{r} &= \mathbf{e}_x x + \mathbf{e}_y y + \mathbf{e}_z z \\ \mathbf{k} &= \mathbf{e}_x k_x + \mathbf{e}_y k_y + \mathbf{e}_z k_z \end{aligned}$$

ϕ is the initial phase, and

$$\mathbf{k} \cdot \mathbf{r} = k_x x + k_y y + k_z z$$

In these expressions $\mathbf{k}$ represents the wavenumber vector, which points in the direction of propagation and

$$k = \sqrt{k_x^2 + k_y^2 + k_z^2}\ ,\ k = \omega / c\ .$$

The inverse of the frequency is called the wave's *period*:

$$T = \frac{1}{f} = \frac{2\pi}{\omega} \tag{4.6}$$

The surfaces in which the phase of $p_h(x, y, z, t)$ is constant are called the *wavefronts*; the perpendiculars to the wavefront are called the *propagation rays*. In the case of a plane wave the wavefronts correspond, obviously, to planes perpendicular to $\mathbf{k}$ (see Fig. 4.1).

The wavenumber allows to introduce the concept of *wavelength*, defined as

$$\lambda = \frac{2\pi}{k} = \frac{c}{f} \tag{4.7}$$

In order to have an idea of the typical values of λ in Underwater Acoustics we can use the reference value $c = 1500$ m/s to obtain 20 Hz and 20 kHz that λ corresponds, respectively, to 75 m and 7,5 cm. The concept of wavelength is valid for all types of

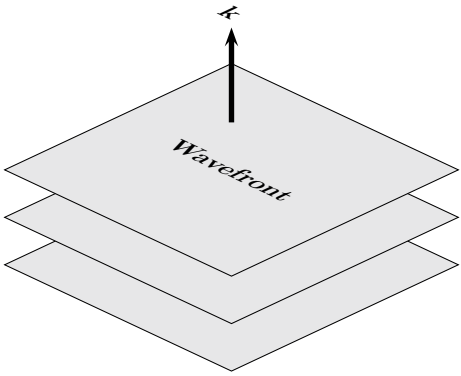


Fig. 4.1 Wavefronts of a plane wave propagating in the positive direction of the Z-axis

waves. The Python Code Snippet 4.1 allows to generate the animation of a plane wave; the direction of propagation can be modified changing the values of the parameter `phi`.

The expressions involving the factor $\mathbf{k} \cdot \mathbf{r} + \omega t$ are also plane waves, which propagate in the opposite direction to the ones with the factor $\mathbf{k} \cdot \mathbf{r} - \omega t$. The superposition of two waves with the same amplitude but the two opposite factors can lead to the generation of a *stationary wave*, in contrast with the original waves, which are *propagating waves*. In the sections that follow and taking into account the observable behaviour of waves we will consider only solutions with a negative sign between $\mathbf{k} \cdot \mathbf{r}$ and ωt.

Code Snippet 4.1 Animation of a plane wave

```
from numpy import *
from matplotlib.pyplot import *
from matplotlib import animation
from mpl_toolkits.mplot3d.axes3d import *

rcParams.update({"text.usetex": True})

def data_gen(framenumber, p, plot):
    wxt = w*t[framenumber]
    p = real( exp( 1j*( kx*X + ky*Y - wxt ) ) )
    ax.clear()
    plot = ax.plot_wireframe(X, Y, p, color='b')
    xlabel(r'$X$',fontsize=18)
    ylabel(r'$Y$',fontsize=18)
    ax.set_zlabel(r'$p_h(x,y,t)$',fontsize=18)
    return plot,

frames   =  36
nperiods =   6
nx       =  51

freq  =   50.0; T = 1.0/freq; w = 2.0*pi*freq
c     = 1500.0
```

```
k      = w/c
# Propagation angle (change to see the effect):
phi    = 0.0
# Wavenumber components:
kx     = k*cos( phi*pi/180.0 )
ky     = k*sin( phi*pi/180.0 )
# Wavelength:
L      = 2.0*pi/k
tmin   =  0
tmax   =  T*nperiods
xmax   = 3.0*L
x = linspace(-xmax,xmax,nx)
y = linspace(-xmax,xmax,nx)

X,Y = meshgrid(x,y);

t = linspace(tmin,tmax,frames)

# Define initial parameters:
p = real( exp( 1j*( kx*X + ky*Y - w*t[0] ) ) )
fig = figure(1)
ax = fig.add_subplot(111, projection="3d")
ax.set_position([0, 0, 0.95, 1])
plot = ax.plot_wireframe(X, Y, p)

# Animation:
animation = animation.FuncAnimation(fig,
data_gen, fargs=(p, plot),
frames=frames,interval=1, blit=False)

show()
```

4.3.2 Cylindrical Waves

In the case of a cylindrical coordinate system (r, z, ϕ) (see Fig. 4.2) one would have that

$$\nabla = \mathbf{e}_r \frac{\partial}{\partial r} + \mathbf{e}_\phi \frac{1}{r} \frac{\partial}{\partial \phi} + \mathbf{e}_z \frac{\partial}{\partial z}$$

which allows to write the homogeneous form of the wave equation as

$$\frac{1}{r} \frac{\partial}{\partial r} \left(r \frac{\partial p_h}{\partial r} \right) + \frac{1}{r^2} \frac{\partial^2 p_h}{\partial \phi^2} + \frac{\partial^2 p_h}{\partial z^2} - \frac{1}{c^2} \frac{\partial^2 p_h}{\partial t^2} = 0$$

A *cylindrical* wave is a wave created by a source, which is extended uniformly along a given direction; if such direction coincides with the Z-axis the axial symmetry of the wave allows to rewrite the previous expression as

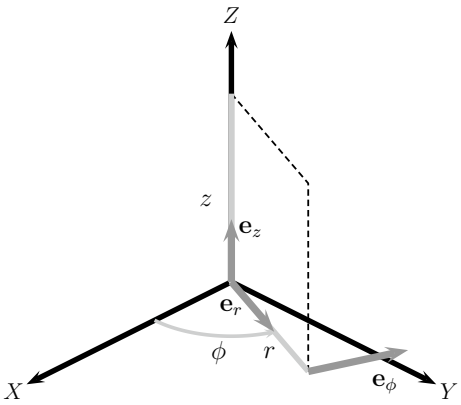


Fig. 4.2 Cylindrical coordinate system.eps

$$\frac{1}{r}\frac{\partial p_h}{\partial r}\left(r\frac{\partial p_h}{\partial r}\right) - \frac{1}{c^2}\frac{\partial^2 p_h}{\partial t^2} = 0$$

which has the *approximated* solution

$$p_h(r, z, t) = \frac{p_0}{\sqrt{r}} \exp\left[i\,(kr - \omega t + \phi)\right] \tag{4.8}$$

valid under the condition that $kr \gg 1$. The Python Code Snippet 4.2 allows to generate the animation of a cylindrical wave in the (X, Y)-plane from the expression given by Eq. (4.8); the animation shows how the waves expand progressively from a common centre; exchanging the negative sign with a positive sign between kr and ωt would produce waves *converging* at the common centre, something that obviously is not observed in real conditions and justifies our choice of the negative sign for the cases of practical interest.

Code Snippet 4.2 Animation of a cylindrical wave

```
from numpy import *
from matplotlib.pyplot import *
from matplotlib import cm
from matplotlib import animation
from mpl_toolkits.mplot3d.axes3d import *

rcParams.update({"text.usetex": True})

def data_gen(framenumber, p, plot):
    f = k*R - w*t[framenumber]
    p = real( exp( 1j*f )/sqrt(R) )
    ax.clear()
    plot = ax.plot_wireframe(X, Y, p, color='b')
    xlabel(r'$X$',fontsize=18)
```

```
    ylabel(r'$Y$',fontsize=18)
    ax.set_zlabel(r'$p_h(r,t)$',fontsize=18)
    ax.set_zlim(-0.5,0.5)
    return plot,

frames  =  36
nperiods =   6
nx       =  51

freq  =   50.0; T = 1.0/freq; w = 2.0*pi*freq
c     = 1500.0
k     = w/c
# Wavelength:
L     = 2.0*pi/k
tmin  =  0
tmax  =  T*nperiods
xmax  = 3.0*L
x = linspace(-xmax,xmax,nx)
y = linspace(-xmax,xmax,nx)

X,Y = meshgrid(x,y)
R = sqrt( X*X + Y*Y )
R[24:26,24:26] = NaN # avoid R = 0...

t = linspace(tmin,tmax,frames)

# Define initial parameters:
f = k*R - w*t[0]
p = real( exp( 1j*f )/sqrt(R) )
fig = figure(1)
ax = fig.add_subplot(111, projection="3d")
ax.set_position([0, 0, 0.95, 1])
plot = ax.plot_wireframe(X, Y, p)

# Animation:
animation = animation.FuncAnimation(fig,
data_gen, fargs=(p, plot),
frames=frames,interval=1, blit=False)

show()
```

4.3.3 Spherical Waves

For a spherical coordinate system (r, ϕ, θ) (see Fig. 4.3) one would have that

$$\nabla = \mathbf{e}_r \frac{\partial}{\partial r} + \mathbf{e}_\phi \frac{1}{r} \frac{\partial}{\partial \phi} + \mathbf{e}_\theta \frac{1}{r \sin \phi} \frac{\partial}{\partial \theta}$$

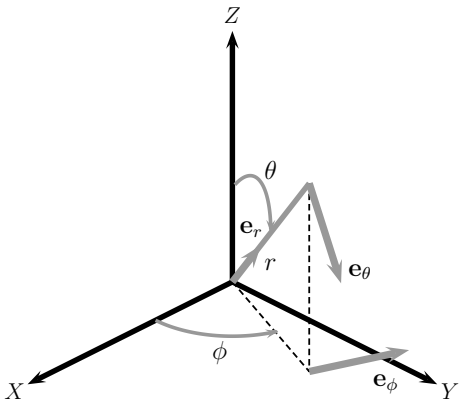


Fig. 4.3 Spherical coordinate system

allowing to write the homogeneous wave equation as

$$\frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial p_h}{\partial r}\right)+\frac{1}{r^2\sin\phi}\frac{\partial}{\partial\phi}\left(\sin\phi\frac{\partial p_h}{\partial\phi}\right)+\frac{1}{r^2\sin^2\phi}\frac{\partial^2 p_h}{\partial\theta^2}-\frac{1}{c^2}\frac{\partial^2 p_h}{\partial t^2}=0$$

An *spherical* wave is a wave created by a *point* source (see Fig. 4.4); under such conditions we can assume that p_h is independent of ϕ and of θ, transforming the previous equation into

$$\frac{1}{r^2}\frac{\partial p_h}{\partial r}\left(r^2\frac{\partial p_h}{\partial r}\right)-\frac{1}{c^2}\frac{\partial^2 p_h}{\partial t^2}=0$$

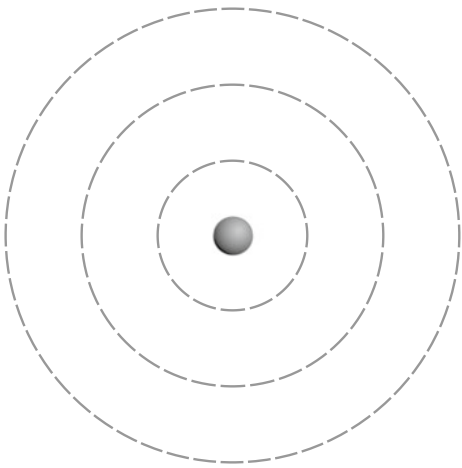


Fig. 4.4 Idealization of a point source emitting spherical waves

The corresponding solution is given by

$$p_h(r,t) = \frac{p_0}{r} \exp\left[i\left(kr - \omega t + \phi\right)\right] \tag{4.9}$$

which unlike the expression for the cylindrical wave is an *exact* solution. The Python Code Snippet 4.3 allows to generate the animation of a spherical wave in the (X, Y)-plane; at a first sight it might seem that the propagation of the spherical wave coincides with the propagation of the cylindrical wave; on closer inspection one can verify that the spherical wave decays in amplitude much faster than the cylindrical wave.

Code Snippet 4.3 Animation of a spherical wave

```
from numpy import *
from matplotlib.pyplot import *
from matplotlib import cm
from matplotlib import animation
from mpl_toolkits.mplot3d.axes3d import *

rcParams.update({"text.usetex": True})

def data_gen(framenumber, p, plot):
    f = k*R - w*t[framenumber]
    p = real( exp( 1j*f )/R )
    ax.clear()
    plot = ax.plot_wireframe(X, Y, p, color='b')
    xlabel(r'$X$',fontsize=18)
    ylabel(r'$Y$',fontsize=18)
    ax.set_zlabel(r'$p_h(r,t)$',fontsize=18)
    ax.set_zlim(-0.2,0.2)
    return plot,

nframes  =  36
nperiods =   6
nx       =  51

freq  =   50.0; T = 1.0/freq; w = 2.0*pi*freq
c     = 1500.0
k     = w/c
# Wavelength:
L     = 2.0*pi/k
tmin  =  0
tmax  =  T*nperiods
xmax  = 3.0*L
x = linspace(-xmax,xmax,nx)
y = linspace(-xmax,xmax,nx)

X,Y = meshgrid(x,y)
R = sqrt( X*X + Y*Y )
R[24:26,24:26] = NaN # avoid R = 0...

t = linspace(tmin,tmax,nframes)

# Define initial parameters:
```

```
f = k*R - w*t[0]
p = real( exp( 1j*f )/R )
fig = figure(1)
ax = fig.add_subplot(111, projection="3d")
ax.set_position([0, 0, 0.95, 1])
plot = ax.plot_wireframe(X, Y, p)

# Animation:
animation = animation.FuncAnimation(fig,
data_gen, fargs=(p, plot),
frames=nframes,interval=1, blit=False)

show()
```

4.3.4 Sommerfeld's Identity

Sommerfeld's identity is an expression found initially in problems of Electromagnetism, but still valid for all problems of wave propagation. The identity defines the expansion of a spherical wave onto an infinite basis of cylindrical waves as

$$\frac{1}{r}\exp(ikr) = 4\pi i \int_{-\infty}^{\infty} J_0(\underline{k}\,\underline{r})\ \exp(ik_z|z|)\ d\underline{k} \tag{4.10}$$

where

$$r = \sqrt{x^2+y^2+z^2} \quad , \quad \underline{r} = \sqrt{x^2+y^2}$$

$$k = \sqrt{k_x^2+k_y^2+k_z^2} \quad , \quad \underline{k} = \sqrt{k_x^2+k_y^2}$$

and $J_0(x)$ represents the zeroth-order Bessel function.

4.4 Phase Velocity, Group Velocity

In the cases discussed in the previous sections we had been dealing with monochromatic waves. For a given value of ω and k we can state that the wave propagates with speed $c = \omega/k$, but if we add several waves with different ω and different k the relationship ω/k will not be the same for different waves. In order to clarify this issue let us consider the superposition of two monochromatic waves that propagate along the X-axis, and have similar ω and k:

$$\begin{aligned} p_h(x,t) &= \cos[(k-\Delta k)\,x - (\omega-\Delta\omega)\,t] \\ &\quad + \cos[(k+\Delta k)\,x - (\omega+\Delta\omega)\,t] \end{aligned}$$

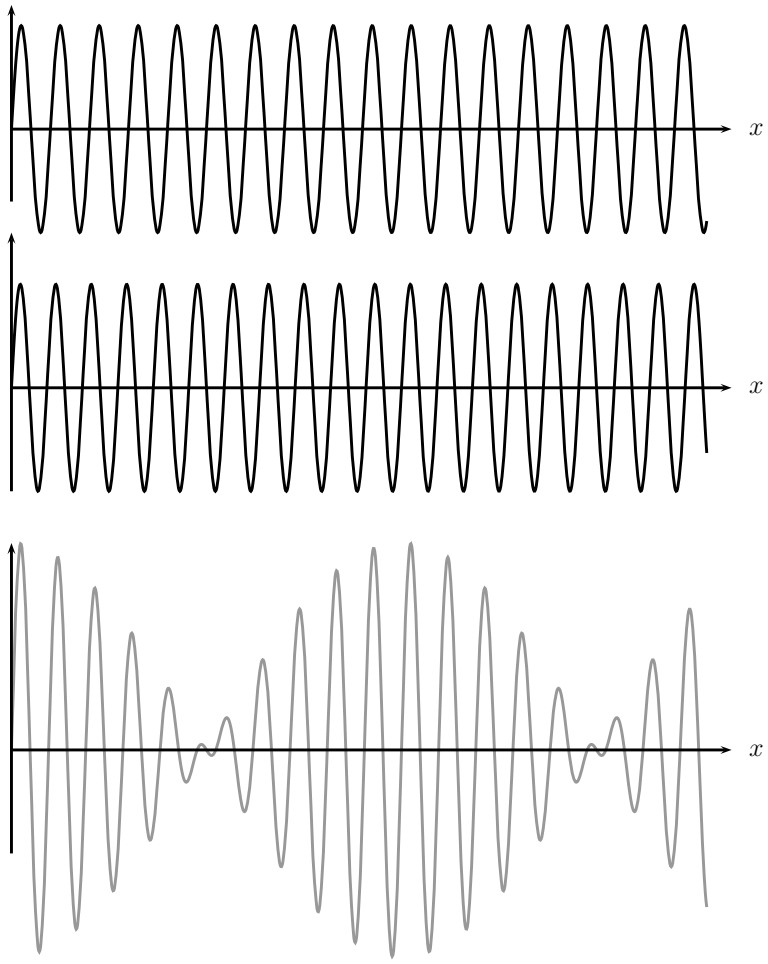


Fig. 4.5 Monochromatic waves with similar ω and k and corresponding superposition

The sum of the trigonometric functions can be rewritten as

$$p_h(x,t) = 2A\cos(kx - \omega t) \ ,$$

where

$$A = \cos(\Delta k\, x - \Delta\omega\, t)$$

The amplitude A involves the plane wave $\cos(kx - \omega t)$, modulating it with a "slow" temporal variation, which imposes restrictions to the speed of $p_h(x,t)$ (see Fig. 4.5).

Taking into account the periodicity of $p_h(x,t)$ we can obtain its speed applying the relationship

$$\Delta x\, k - \Delta\omega\, t = (k + \Delta k)\, \Delta x - (\omega + \Delta\omega)\, \Delta t$$

which would allow to obtain that

$$v = \frac{\Delta x}{\Delta t} = \frac{\Delta\omega}{\Delta k}$$

This expression represents the *group velocity*, that is, it represents a *mean* value of the speed of propagation of $p_h(x,t)$. In order to generalize this result to the case of the superposition of many monochromatic waves we need to consider the limit when $\Delta k \to 0$, allowing to obtain that

$$\text{group velocity} = \lim_{\Delta k \to 0} \frac{\Delta\omega}{\Delta k} = \frac{d\omega}{dk}$$

On another side the expression ω/k represents the *phase velocity* of $p_h(x,t)$. The relevance of this distinction between velocities resides in the fact that the *energy* of a group of waves propagates with group velocity and not with phase velocity.

The Python Code Snippet 4.4 allows to generate the animation of a group of two waves; it is strongly recommended to execute the code changing the parameter that defines the phase velocity to see the behaviour of the group in different situations.

Code Snippet 4.4 Phase velocity and group velocity

```
from numpy import *
from matplotlib.pyplot import *
from matplotlib import animation

rcParams.update({"text.usetex": True})

def data_gen(framenumber, p, plotxp):
    p1 = exp( 1j*( k1*x - w1*t[framenumber] ) )
    p2 = exp( 1j*( k2*x - w2*t[framenumber] ) )
    p = 0.5*real( p1 + p2 )
    ax.clear()
    plot(x,p)
    xlabel(r'$x$',fontsize=18)
    ylabel(r'$p(x,t)$',fontsize=18)
    grid(True)
    return plotxp,

nframes  = 101
nperiods =  50
nx       = 201

v  = 1500.0 # vphase
# Uncomment and run the animation:
c  = v # vgroup = vphase
#c = 0 # vgroup = 0
#c = -250 # vgroup < 0
```

```
f1  = 50.0
w1 = 2.0*pi*f1
k1 = w1/v; dk = k1/10.0
k2 = k1 + dk
dw = c*dk
w2 = w1 + dw
f2 = w2/( 2.0*pi )
T1 = 1.0/f1
T2 = 1.0/f2

print(v,c)

L1 = 2.0*pi/k1
L2 = 2.0*pi/k2

tmin = 0.0
tmax = max([T1,T2])*nperiods

xmax = 10.0*max([L1,L2])

x = linspace(-xmax,xmax,nx)
t = linspace(tmin,tmax,nframes)

# Define initial parameters:
p1 = exp( 1j*( k1*x - w1*t[0] ) )
p2 = exp( 1j*( k2*x - w2*t[0] ) )
p = 0.5*real( p1 + p2 )
fig = figure(1)
ax = fig.gca()
plotxp = plot(x,p)

# Animation:
animation = animation.FuncAnimation(fig,
data_gen, fargs=(p, plotxp),
frames=nframes,interval=1, blit=False)

show()
```

4.5 Attenuation

The analytical solutions for cylindrical and spherical waves indicate that the wave amplitude decays progressively during propagation; this is called the *geometric* decay. Yet, in real conditions, propagation also exhibits a *dissipative* behaviour, that is, there is a phenomenon of *attenuation* that is independent of the geometric decay; attenuation means that part of the wave's energy is going to be transmitted to the fluid because of the fluid's viscosity and chemical composition; the degree of attenuation depends fundamentally on the signal's frequency, and also on the fluid's depth, density and temperature. In the case of salt water, for instance, the wave atten-

uation is related to processes of chemical relaxation of the components of marine salt, such as $MgSO_4$, $B(OH)_3$ and $MgCO_3$.

In Underwater Acoustics the attenuation is characterized mathematically by the expression

$$I = I_0\ 10^{-\alpha r/10} \tag{4.11}$$

where I represents the wave's intensity, I_0 is the initial intensity, r is the propagation distance and α is the *attenuation coefficient*; in Eq. (4.11) the dimensions of α correspond to dB/m. For acoustic pressure one would write that

$$p = p_0\ 10^{-\alpha r/20} \tag{4.12}$$

Attenuation models in Underwater Acoustics are obtained empirically relying on functional expressions, which are fitted to experimental data; such models have little to do with theoretical models of attenuation; one of the simplest models is the Thorp model, which depends only on frequency, and is given by [3]:

$$\alpha(f) = 3,3 \times 10^{-3} + \frac{0,11 f^2}{1+f^2} + \frac{44 f^2}{4100+f^2} + 3 \times 10^{-4} f^2 \tag{4.13}$$

On the other side, according to Francois and Garrison model the attenuation coefficient is given by [22, 23]:

$$\alpha(f) = \frac{A_1 f_1 f^2}{{f_1}^2 + f^2} + \frac{A_2 P_2 f_2 f^2}{{f_2}^2 + f^2} + A_3 P_3 f^2 \tag{4.14}$$

where

$$\begin{aligned} A_1 &= \frac{8.86}{c} \times 10^{(0.78pH-5)}\ \text{dB/(km} \cdot \text{kHz)} \\ f_1 &= 2.8\,(S/35)^{1/2}\ 10^{(4-1245/\theta)}\ \text{kHz} \end{aligned}$$

c is the propagation speed, given by

$$c = 1412 + 3.21\,T + 1.19\,S + 0.0167D$$

in which T represents the water temperature in degree Celsius, $\theta = 273 + T$, S is the salinity in parts per thousand and D is water depth in metres; additionally

$$\begin{aligned} A_2 &= \frac{21.44S}{c}(1 + 0.025T)\ \text{dB/(km} \cdot \text{kHz)} \\ P_2 &= 1 - 1.37 \times 10^{-4} D + 6.2 \times 10^{-9} D^2 \\ f_2 &= \frac{8.17 \times 10^{(8-1990/\theta)}}{1 + 0.0018(S-35)}\ \text{kHz} \end{aligned}$$

for $T \leq 20°\text{C}$ one would have that

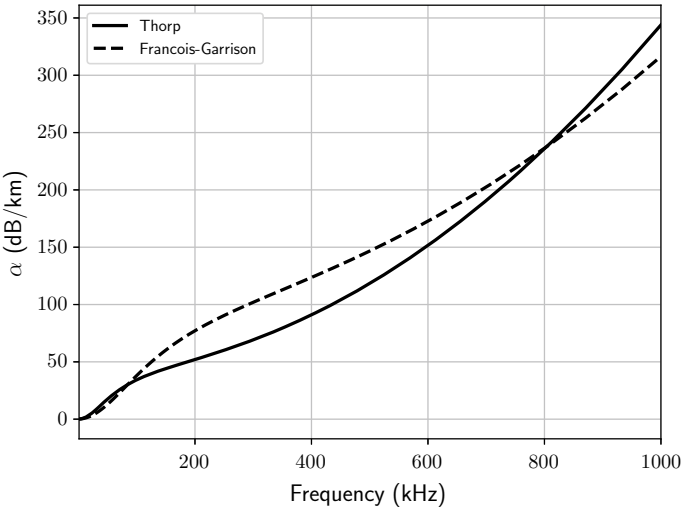


Fig. 4.6 Acoustic attenuation models

$$A_3 = 4.937 \times 10^{-4} - 2.59 \times 10^{-5} T + 9.11 \times 10^{-7} T^2 - 1.50 \times 10^{-8} T^3 \text{ dB}/(\text{km} \cdot \text{kHz}^2)$$

while for $T > 20°\text{C}$

$$A_3 = 3.964 \times 10^{-4} - 1.146 \times 10^{-5} T + 1.45 \times 10^{-7} T^2 - 6.50 \times 10^{-10} T^3 \text{ dB}/(\text{km} \cdot \text{kHz}^2)$$

In Eqs. (4.13) and (4.14) the frequency should be given in kHz and the dimensions of α correspond to dB/km. Figure 4.6, generated by the Python Code Snippet 4.5, compares the two models for a common set of frequencies (see the parameters of Eq. (4.14) in the code); the comparison shows that the two models produce slightly different values of the attenuation coefficient, but in both cases α increases progressively with frequency.

Code Snippet 4.5 Thorp and Francois-Garrison acoustic attenuation models

```
from numpy import *
from matplotlib.pyplot import *

rcParams.update({"text.usetex": True})

def francoisgarrison(freq=None,
    T=None,S=None,pH=None,z=None):

    c    = 1412.0 + 3.21*T + 1.19*S + 0.0167*z
```

```
    theta = 273.0 + T
    fxf = freq**2
    f1 = 2.8*sqrt(S/35.0)*10**(4.0-1245.0/theta)
    f2 = 8.17*10**(8.0-1990.0/theta)/ \
         ( 1.0 + 0.0018*(S-35.0) )
    A1 = 8.86/c*10**(0.78*pH-5)
    A2 = 21.44*S/c*( 1.0 + 0.025*T )
    if T <= 20:
       A3 = 4.937e-4 - 2.59e-5*T + \
            9.11e-7*T**2 - 1.50e-8*T**3
    else:
       A3 = 3.964e-4 - 1.146e-5*T + \
            1.45e-7*T**2 - 6.5e-10*T**3
    P2 = 1 - 1.35e-4*z + 6.2e-9*z**2
    P3 = 1 - 3.83e-5*z + 4.9e-10*z**2

    a = A1*f1*fxf/( f1**2 + fxf ) + \
        A2*P2*f2*fxf/( f2**2 + fxf ) + \
        A3*P3*fxf

    return a

nfreqs = 101
fmin =     1
fmax =  1000
n = linspace(0,3,101)
freqs = 10**n # kHz!
S  = 35.0
T  = 20.0
pH =  8.0
Z  =  0.0
fxf = freqs**2

aT = 3.3e-3 + 0.11*fxf/( 1.0 + fxf ) + \
     44.0*fxf/( 4100.0 + fxf ) + \
     3.0e-4*fxf
aFG = francoisgarrison(freqs,T,S,pH,Z)

figure(1)
plot(freqs,aT ,'k'  ,linewidth=2,
label='Thorp')
plot(freqs,aFG,'k--',linewidth=2,
label='Francois-Garrison')
xlabel('Frequency (kHz)',fontsize=18)
ylabel(r'$\alpha$ (dB/km)',fontsize=18)
xlim(1,1000)
legend(loc=2)
grid(True)

show()
```

4.6 The Doppler Effect

The Doppler effect corresponds to the change in frequency of a propagating wave, due to the relative motion between the source and the receiver. From the point of the receiver the signal frequency *increases* when the source approaches the receiver and *decreases* in the opposite case.

In order to determine the frequency change let us consider a source that emits waves with a period T. Each time a wave is emitted the propagation of this wave is supported by the propagation medium, whose elastic properties define the value of the speed of propagation c. Although the waves are emitted periodically by the source their speed of propagation is *independent* of the state of motion of the source. The same does not apply to the wavelength. Let us consider, for instance, a point source emitting circular waves: for a static source the wavefronts are all located at the same intervals, meaning that the wavelength *is the same* independently of the direction in which the waves arrive at a static receiver. Therefore, from the point of view of such receiver the wavelength corresponds to

$$f_0 = \frac{c}{\lambda_0}$$

where the zeroth index indicates the frequency measured by the receiver when the source is static.

Let us consider now the case in which the source approaches a receiver 1 with a velocity v_s (see Fig. 4.7). Each wavefront remains centred at the source position, but for a moving source the wavefronts are tighter in the direction of motion and distance

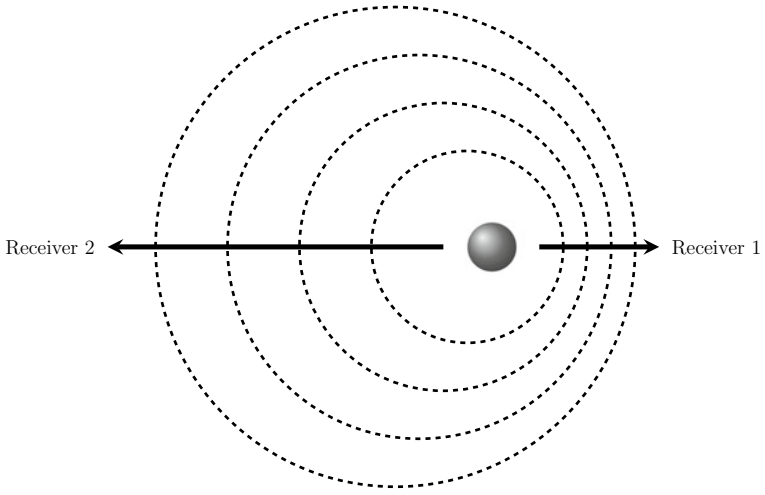


Fig. 4.7 Wavefronts of a moving point source

from each other in the opposite direction. From the point of view of receiver 1 the wavelength becomes

$$\lambda_1 = (c - v_s)\, T < \lambda_0$$

It then becomes obvious that for this receiver the wave frequency corresponds to

$$f_1 = \frac{c}{\lambda_1} = \frac{c}{(c - v_s)\, T} = \frac{c}{(c - v_s)} f_0$$

where we took into account that $f_0 = 1/T$. We can now conclude that for the receiver 1 the variation of the wave frequency is given by

$$\Delta f = f_1 - f_0 = \frac{c}{(c - v_s)} f_0 - f_0 = f_0 \left[\frac{c}{(c - v_s)} - 1 \right]$$

which can be rewritten as

$$\Delta f = f_0 \frac{v_s}{c - v_s}$$

that is obviously positive. In other words, for the receiver 1 the frequency of the approaching source *increases* relative to what this receiver would measure for a static source. For a receiver 2, located at the opposite side of the source (relative to receiver 1), the reasoning is similar, but the conclusion is different, namely, that the wavelength increases instead of decreasing:

$$\lambda_2 = (c + v_s)\, T > \lambda_0$$

implying that

$$f_2 = \frac{c}{\lambda_2} = \frac{c}{(c + v_s)} f_0$$

Therefore, for the receiver 2 the frequency change of the wave emitted by the distancing source corresponds to

$$\Delta f = f_2 - f_0 = \frac{c}{(c + v_s)} f_0 - f_0 = f_0 \left[\frac{c}{(c + v_s)} - 1 \right]$$

meaning that

$$\Delta f = -f_0 \frac{v_s}{c + v_s}$$

this result indicates that for the receiver 2 the wave frequency *decreases*.

The previous discussion did not include the case in which the receiver and the source both move at the same time, but the fact remains that the frequency measured by the receiver will be different from the frequency measured in a static case. We would skip the technical details of both source and receiver moving and indicate that the frequency in this case is given by the expression

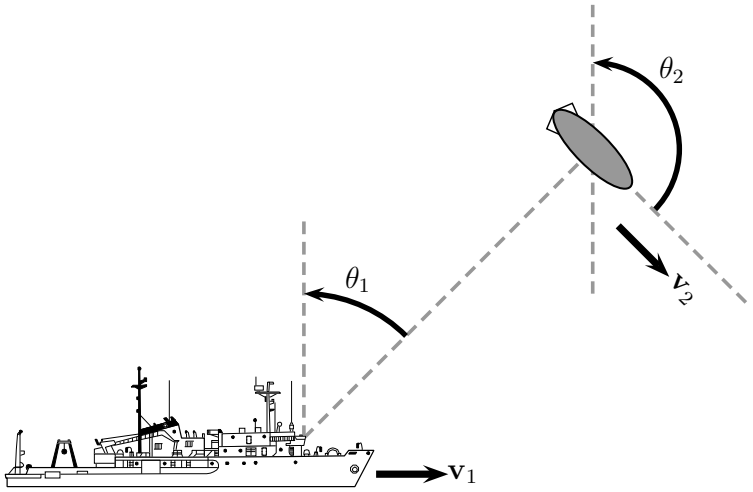


Fig. 4.8 Velocities and directions of the vessel and AUV

$$f = \left(\frac{c + v_r}{c - v_s}\right) f_0 \tag{4.15}$$

where

- v_r represents the receiver velocity relative to the propagation medium; it should be considered *positive* if the receiver *approaches* the source and negative in the opposite case.
- v_s represents the source velocity relative to the propagation medium; it should be considered *positive* if the source *approaches* the receiver and negative in the opposite case.

For the case $v_s = 0$ and $v_r = v$ we can write in general that

$$\Delta f = 2\left(\frac{v}{c}\right) f_0 \tag{4.16}$$

Equation (4.15) can be easily adapted to the case in which velocities are not aligned along a common direction. Let us consider the following example: a research vessel, moving with a velocity $v_1 = 30$ knots[1] is locating an *AUV* (*Autonomous Underwater Vehicle*), which is moving with a velocity $v_2 = 5$ knots. To accomplish such task the vessel transmits a burst with a frequency of 5 kHz, which is reflected by the AUV and is received as an echo back at the vessel. The *Line of Sight* (abbreviated as *LOS*) corresponds to angles of $\theta_1 = 75°$ and $\theta_2 = 120°$ (see Fig. 4.8).

In order to calculate the echo frequency we need to project the velocities along the LOS:

[1] 1 knot = 0.514 m/s.

$$V_1 = v_1 \sin\theta_1 = 0.514 \times 30 \times \sin(75°) = 14.89 \text{ m/s}$$

and

$$V_2 = v_2 \cos(\theta_2 - \theta_1) = 0.514 \times 5 \times \cos(120° - 75°) = 1.82 \text{ m/s}$$

Now, we must apply Eq. (4.15) not one, but two times: one for the transmitting vessel, where V_1 represents the velocity of a source approaching a receiver, which is moving with velocity V_2; when the echo returns to the vessel V_1 will now represent the velocity of a receiver, which is approaching a source, and this source is distancing from the receiver with velocity V_2. Merging the two applications of Eq. (4.15) into a single expression we can write that the frequency of the echo will correspond to

$$f' = 5000 \times \left(\frac{1500 + (-1.82)}{1500 - 14.89}\right) \times \left(\frac{1500 + 14.89}{1500 - (-1.82)}\right) = 5088 \text{ Hz}$$

As shown by Eq. (4.16) the state of motion of a source leads always to a frequency change; this change implies a temporal distribution of frequencies in the received signal, which differs from the distribution of frequencies of the original signal, implying that the signal's shape is not preserved in the reception. The change can be mitigated (or eventually eliminated) transmitting a signal $s[\phi(t)]$ such that

$$f(t) = f_r(t - t_0) \tag{4.17}$$

where $f = d\phi/dt$, $f_r(t)$ represents the phase of the received signal and t_0 is a constant; the signals that satisfy Eq. (4.17) are called *Doppler invariant*. Let us explore this issue in more detail: for a signal with linear modulation of frequency we can write that

$$\phi(t) = 2\pi\left(f_0 t + \frac{1}{2}\kappa t^2\right)$$

leading to the expression [24]

$$t_0 = \frac{\kappa t\left(s^2 - 1\right) + f_0(s - 1)}{\kappa s^2} - \tau$$

in which s stands for the *Doppler factor*, given by

$$s = \frac{c - v}{c + v}$$

and τ is the signal's delay, which for a distance D between the source and the receiver corresponds to

$$\tau = \frac{2D}{c}$$

The expression for t_0 shows that it depends linearly on time; therefore, a linear modulation of frequency cannot be used to satisfy Eq. (4.17). Since Δf depends linearly on the velocity it makes sense to consider a *hyperbolic* modulation of frequency, that is, a modulation which is inversely proportional to the signal's duration. In fact, for the signal

$$s(t) = \cos\left[\phi(t)\right]$$

where

$$\phi(t) = \frac{2\pi}{\kappa} \log\left(1 + \kappa f_0 t\right)$$

with a frequency modulation given by

$$f(t) = \frac{1}{\kappa t + 1/f_0}$$

and

$$f_r(t) = \frac{1}{\kappa\left(t - \tau\right) + 1/\left(s f_0\right)}$$

It follows from these expressions that [24]

$$t_0 = \frac{1 - s}{\kappa s f_0} - \tau$$

meaning that t_0 is independent of time. This result shows that with a proper choice of parameters the hyperbolic modulation of frequency can be used for the design of signals with Doppler invariance.

Chapter 5
Waves in Bounded Media

5.1 Introduction

The discussion presented in Chap. 4 described the problem of wave propagation in a fluid in the absence of boundaries, and for the case of a constant speed of propagation. The discussion will be extended in this chapter considering again wave propagation, but in elastic media, including wave refraction due to the space variability of the speed of propagation, and also wave reflection due to the existence of boundaries between media with different properties.

5.2 The Navier–Stokes Equation

The starting point for the discussion of the equivalent of Newton's second law of motion in the case of elastic media is given by the *Navier–Stokes equation* [25]:

$$\rho \frac{\partial^2 \mathbf{u}}{\partial t^2} = (\lambda + 2\mu)\, \nabla (\nabla \cdot \mathbf{u}) - \mu \nabla \times (\nabla \times \mathbf{u}) + \mathbf{f} \tag{5.1}$$

where

- $\mathbf{u}$ represents the *particle displacement*,
- λ and μ are the *Lamé parameters*,
- $\mathbf{f}$ represents the external force, and
- ∇ represents the *nabla* operator.

Let us consider the case when the displacement can be considered as the gradient of a scalar potential:

$$\mathbf{u} = \nabla \Phi$$

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023

O. C. Rodríguez, *Fundamentals of Underwater Acoustics*,
https://doi.org/10.1007/978-3-031-31319-6_5

which satisfies automatically the condition

$$\nabla \times (\nabla \Phi) = 0$$

From the point of view of Vector Calculus this expression reflects the fact that $\nabla\Phi$ and ∇ are"aligned" along a common direction, therefore their outer product is zero; from the point of Physics the expression means that $\mathbf{u}$ has a "radial" behaviour between equipotential surfaces. In order to deal only with scalar quantities we can now represent the external force as the gradient of another potential

$$\mathbf{f} = -\nabla F$$

Dimensional analysis allows to find that F has units of pressure, and that justified the formality of the negative sign.[1] After substitution of the scalars Φ and F into Eq. (5.1) we can obtain a wave equation for the potential Φ:

$$\nabla^2 \Phi - \frac{1}{c_p^2} \frac{\partial^2 \Phi}{\partial t^2} = \frac{F}{\rho c_p^2}$$

in which the speed of propagation of the potential Φ is given by the expression

$$c_p = \sqrt{\frac{\lambda + 2\mu}{\rho}} \tag{5.2}$$

this velocity is known as *compressional speed* [26].

Let us consider now the case when $\mathbf{u}$ can be considered as the rotational of a vector potential

$$\mathbf{u} = \nabla \times \boldsymbol{\Psi}$$

which satisfies the condition

$$\nabla \cdot (\nabla \times \boldsymbol{\Psi}) = 0$$

From the point of view of Vector Calculus this expression indicates that $\nabla \times \Psi$ and ∇ are "orthogonal" to each other, and thus their inner product is zero; from the point of view of Physics the expression means that the displacement trajectories are "curled" between equipotential vector fields. After substitution of the vector potential into the homogeneous form of Eq. (5.1) we can obtain a wave equation for the potential $\boldsymbol{\Psi}$:

$$\nabla^2 \boldsymbol{\Psi} - \frac{1}{c_s^2} \frac{\partial^2 \boldsymbol{\Psi}}{\partial t^2} = 0$$

[1] In the mechanics of elastic media instead of pressure one needs to deal with the *stress* tensor, which is composed of normal and shear components; the opposite of the normal components corresponds to pressure.

where the speed of propagation corresponds now to

$$c_s = \sqrt{\frac{\mu}{\rho}} \tag{5.3}$$

this velocity is called the *shear speed*. Combining Eq. (5.2) with Eq. (5.3) we can obtain now for the general case that

$$\frac{c_p}{c_s} = \sqrt{\frac{\lambda + 2\mu}{\mu}} > 1 \tag{5.4}$$

It is well known that the problem of finding a vector potential for a given displacement has no unique solution, but this problem can be simplified by resorting to the symmetry of the coordinate system. For instance, in a system with cylindrical symmetry, the radial and vertical components of displacement can be written as [25, 26]

$$u_r = \frac{\partial \Phi}{\partial r} + \frac{\partial^2 \Psi}{\partial z \partial r}$$

and

$$u_z = \frac{\partial \Phi}{\partial z} - \frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial \Psi}{\partial r}\right)$$

where the scalar functions $\Phi(r, z)$ and $\Psi(r, z)$ satisfy the wave equations

$$\underline{\nabla}^2 \Phi = \frac{1}{c_p^2}\frac{\partial^2 \Phi}{\partial t^2} \quad \text{and} \quad \underline{\nabla}^2 \Psi = \frac{1}{c_s^2}\frac{\partial^2 \Psi}{\partial t^2}$$

in which

$$\underline{\nabla}^2 = \frac{\partial^2}{\partial r^2} + \frac{1}{r}\frac{\partial}{\partial r} + \frac{\partial^2}{\partial z^2}$$

To obtain a wave equation for the pressure field we can write now that

$$p = F - (\lambda + 2\mu)(\nabla \cdot \mathbf{u})$$

After substitution into Eq. (5.1) we can obtain the expression

$$\nabla\left(\frac{1}{\rho}\nabla p\right) - \frac{1}{\rho c_p^2}\frac{\partial^2 p}{\partial t^2} = -\frac{s}{\rho}$$

where

$$s = \frac{1}{c_p^2}\frac{\partial^2 F}{\partial t^2}$$

the above wave equation becomes Eq. (4.1) when the medium density is constant.

The *particle velocity* given by

$$\mathbf{v} = \frac{\partial \mathbf{u}}{\partial t} \tag{5.5}$$

can be related to pressure in the following way: let us consider the case when $\boldsymbol{\Psi} = \mathbf{0}$ and $F = 0$, leading to the expression

$$p = -\rho c_p^2 \, (\nabla \cdot \mathbf{u}) \tag{5.6}$$

Under these conditions Eq. (5.1) becomes

$$\frac{\partial \mathbf{v}}{\partial t} = -\frac{\nabla p}{\rho} \tag{5.7}$$

In the monochromatic case we can write that $p = Pe^{-i\omega t}$ and $\mathbf{v} = \mathbf{V}e^{-i\omega t}$, which transforms Eq. (5.7) into

$$\mathbf{V} = -i\frac{\nabla P}{\rho\omega} \tag{5.8}$$

This expression identifies the particle velocity, in the frequency domain, as the gradient of acoustic pressure. In such context the scalar function

$$\Phi_v = -i\frac{P}{\rho\omega} \tag{5.9}$$

defines a *potential of particle velocity*. The expression given by Eq. (5.8) can be simplified in the case of cylindrical and spherical waves, for which we can write that

$$P = p_0 r^{-n} e^{i(kr-\omega t)}$$

with $n = 1$ for a spherical wave and $n = 1/2$ for a cylindrical wave. Since the waveguide symmetry restricts the solution to depend on r only we can drop the vector notation and write that

$$V = -\frac{i}{\rho\omega}\frac{dP}{dr}$$

After some algebra we can conclude that

$$V = \left(\frac{1}{\rho c} - \frac{in}{\rho\omega r}\right) P$$

which shows that in the case of the *near field* the coefficient of proportionality between V and P is complex. We can obtain an approximation for the *far field* by making $r \to \infty$, leading to the expression [27]

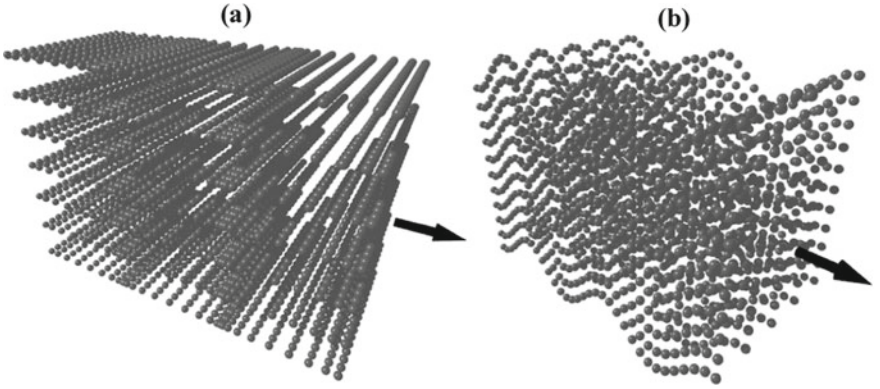


Fig. 5.1 Particle oscillations of an elastic medium for: **(a)** a compressional wave; **(b)** a shear wave. In both cases the arrow indicates the direction of wave propagation

$$V = \frac{P}{\rho c} \tag{5.10}$$

in which the radial component of the particle velocity is nothing more than a "copy" of pressure, weighted by the fluid density and by the velocity of propagation.

The previous discussion is relevant in the sense of a generalization of wave propagation, by noticing that

- An elastic medium can support the propagation of two different types of waves, namely, *compressional* waves and *shear* waves.
- There are two fundamental differences between these two types of waves: in a compressional wave the oscillatory motion of the particles takes place along the direction of wave propagation; in a shear wave the oscillations are *orthogonal* to the direction of propagation (see Fig. 5.1).
- Equation (5.4) shows that in the general case $c_p > c_s$, hence the denomination used in Seismology of *P waves* ("primary") for compressional waves, and *S waves* ("secondary") for shear waves.
- For a fluid $\mu = 0$, and that identifies the solution of Eq. (4.1) as representing a compressional wave.
- The devices that allow to transmit and record pressure waves are called *transducers*; the devices that record the particle velocity are called *vector sensors*.

The Python Code Snippet 5.1 and 5.2, allow to generate animations of a compressional and a shear wave.

Code Snippet 5.1 Animation of a compressional wave.

```
from numpy import *
from matplotlib.pyplot import *
from matplotlib import animation

rcParams.update({"text.usetex": True})

def data_gen(framenumber, p, plotxp):
    p = sin( k*x - w*t[framenumber] )
    ax.clear()
    nx = x.size
    for j in range(nx):
        plot( (x[j]+p[j])*ones(nx),y,'ko')
    xlabel(r'$X$',fontsize=18)
    ylabel(r'$Y$',fontsize=18)
    xlim(-3*pi,3*pi)
    grid(True)
    return plotxp

frames = 101
nx     = 51
ny     = 51
w      = 1.0
k      = 1.0
xmax   = 3.0*pi
ymax   = 1.0
t      = linspace(0.0,10.0*pi,frames)
x      = linspace(-xmax,xmax,nx)
y      = linspace(-ymax,ymax,ny)

# Define initial parameters:
p = sin( k*x - w*t[0] )
fig = figure(1)
ax = fig.gca()
plotxp = plot( x+p,y,'ko')

# Animation:
animation = animation.FuncAnimation(fig,
data_gen, fargs=(p, plotxp),
frames=frames,interval=1, blit=False)

show()
```

Code Snippet 5.2 Animation of a shear wave.

```
from numpy import *
from matplotlib.pyplot import *
from matplotlib import animation

def data_gen(framenumber, p, plotxp):
```

```
    p = sin( k*x - w*t[framenumber] )
    ax.clear()
    plot(x,p,'ko')
    stem(x,p,'k--')
    xlabel(r'$X$',fontsize=18)
    ylabel(r'$Y$',fontsize=18)
    grid(True)
    return plotxp

frames = 101
nx     = 21
w      = 1.0
k      = 1.0
xmax   = 2.0*pi
t      = linspace(0.0,20.0*pi,frames)
x      = linspace(0.0,xmax,nx)

# Define initial parameters:
p = sin( k*x - w*t[0] )
fig = figure(1)
ax = fig.gca()
plotxp = plot(x,p,'ko')

# Animation:
animation = animation.FuncAnimation(fig,
data_gen, fargs=(p, plotxp),
frames=frames,interval=1, blit=False)

show()
```

5.3 Acoustic Impedance

The *impedance* is a concept of extreme importance in Physics; in circuit theory the impedance corresponds to a complex quantity, which in the frequency domain can be associated with each element in the circuit, in the same way as a resistance can be associated with current in the time domain; in another areas the definition of impedance is adapted depending on the context of the given problem. In Underwater Acoustics, in order to introduce the concept of impedance in a coherent manner, it is important to start with the expression for vector intensity, which combines pressure with particle velocity and is given by

$$\mathbf{I} = p\mathbf{v}$$

Since the intensity is often treated as a scalar quantity we need to rewrite the previous equation as

$$I = p\,|\mathbf{v}|$$

This identity should be considered with some care, because it only makes sense in the context of a wave propagating in a given direction, and without missing the fact that p should be considered in absolute value. Let us introduce the impedance as

$$Z = \frac{p}{|\mathbf{v}|}$$

which obviously defines a real quantity for plane waves in the far field. After some algebra we can obtain that

$$I = \frac{p^2}{Z}$$

In order to relate impedance to the medium properties let us consider now, without loss of generality, the case of a plane wave in which

$$\mathbf{u} = u_0 e^{i(kx-\omega t)} \mathbf{e}_x$$

According to Eq. (5.6), and preserving only the absolute value, we can obtain that

$$p = k\rho c^2 u_0$$

On another side, since $\mathbf{v} = \dot{\mathbf{u}}$, it follows that

$$|\mathbf{v}| = u_0 \omega$$

which allows to obtain for the impedance that

$$Z = \frac{k\rho c^2 u_0}{u_0 \omega} = \frac{\rho c^2}{\omega / k} = \rho c$$

Therefore, the intensity and pressure become related through the expression [3, 28]

$$I = \frac{p^2}{\rho c}$$

We can see from this result that in the context of a propagating wave the impedance corresponds to a real quantity; in other cases, like, for instance, a stationary wave, the impedance can correspond to a complex quantity.

5.4 Wave Dispersion

The extreme simplicity of Eq. (4.3) can obscure one of the most important facts relative to the propagation of a group of waves, namely, that such propagation can exhibit a *dispersive* behaviour, that is, that the shape of the group can change progressively as

the group propagates. The change of shape is related to the fact that the propagation medium can impose restrictions to the speed of each group component, and therefore *slower* components will fall behind relative to the *faster* components. In Optics, for instance, the most representative demonstration of light dispersion corresponds to the decomposition of white light into the spectrum of colours after propagating inside a prism; in this specific case light dispersion is related to the dependence of the index of refraction on the wavelength. In Acoustics wave dispersion is equally important, but not that easy to exemplify. Let us consider as a starting point a more general version of Eq. (4.3), namely,

$$|\mathbf{k}| = \frac{\omega}{c} \quad or \quad \sqrt{k_x^2 + k_y^2 + k_z^2} = \frac{\omega}{c} \tag{5.11}$$

For a particular case of wave propagation inside a fluid with a depth D and for a case with cylindrical symmetry (r, z) it is possible to obtain the relationship [29]

$$\underline{k} = \sqrt{\frac{\omega^2}{c^2} - k_z^2}$$

where

$$\underline{k} = \sqrt{k_x^2 + k_y^2} \quad , \quad k_z = \left(m - \frac{1}{2}\right)\frac{\pi}{D}$$

and $m = 1, 2, 3 \ldots$, that is, even in a monochromatic case a wave can propagate as a superposition of components of the type $e^{i\underline{k}_m r}$. The phase speed is given in this case by $v_p = \underline{k}_m/\omega \neq c$, and it is obviously different from component to component.

In Physics, in the general case, the dispersive behaviour of a group of waves is characterized by the dependence between ω and k

$$\omega = \omega(k) \tag{5.12}$$

which depends on the specificity of each problem; Eq. (5.12) is known in the general case as the *dispersion relation*.

5.5 Wave Refraction

The change of direction of a wave is known as *refraction* and it is induced by the space (and time) variation of the velocity of propagation. According to *Fermat's principle* (also known as the *principle of minimal time*) such change takes place in order to minimize the propagation time between two given points. Let us consider the application of this principle in the case of two fluids with the same density, but different propagation speeds c_1 and c_2, separated by a horizontal boundary (see Fig. 5.2). The wave propagates from one point with coordinates (x_1, y_1) to a point

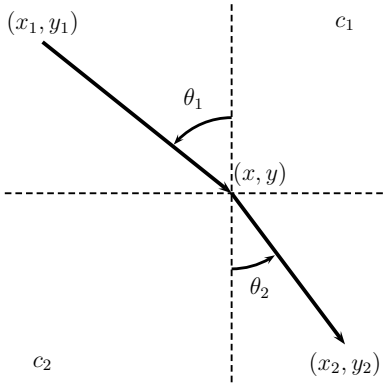


Fig. 5.2 Possible trajectory of a ray propagating between two points

with coordinates (x_2, y_2), exhibiting a change of direction at the point (x, y); it is intended to determine the value of x that minimizes the propagation time between the two points.

In the general case we can see that the propagation time is given by $t(x) = t_1(x) + t_2(x)$, where

$$t_1(x) = \frac{\Delta s_1}{c_1} \quad , \quad t_2(x) = \frac{\Delta s_2}{c_2}$$

with

$$\Delta s_1 = \sqrt{(x - x_1)^2 + (y - y_1)^2} \quad \text{and} \quad \Delta s_2 = \sqrt{(x_2 - x)^2 + (y_2 - y)^2}$$

Solving the equation $dt/dx = 0$ in order to x we can obtain that $t(x)$ is minimal only when the angles θ_1 and θ_2 satisfy the condition

$$c_2 \sin\theta_1 - c_1 \sin\theta_2 = 0 \tag{5.13}$$

which in its classic form is used to be written as

$$\frac{\sin\theta_1}{c_1} = \frac{\sin\theta_2}{c_2} \tag{5.14}$$

This expression is well known as *Snell's law*. In Eq. (5.14) the angle θ_1 is calculated relative to the normal to the boundary that separates the two fluids and it is known as the *angle of incidence*; yet, in Underwater Acoustics, it is more common to consider the *grazing angle* $\underline{\theta}$, which is the complementary to the angle of incidence

$$\underline{\theta} = \frac{\pi}{2} - \theta$$

since the speed of propagation is often a function of depth Snell's law which is then rewritten as

$$\frac{\cos\theta(z)}{c(z)} = \text{constant} \tag{5.15}$$

5.6 The Speed of Sound

The speed of sound propagation in the water column can be measured resorting to velocimeters, which have an accuracy up to 0.3 m/s. Alternatively, one can measure the water temperature T and salinity S for different values of depth z, and calculate the speed of sound using empirical relationships, such as Medwin's formula [30]

$$\begin{aligned} c(z,T,S) = {} & 1449.2 + 4.6\,T - 5.5\times10^{-2}T^2 + 2.94\times10^{-4}T^3 \\ & + (1.34 - 0.01\,T)\,(S-35) \\ & + 1.6\times10^{-2}z \end{aligned} \tag{5.16}$$

or Mackenzie's formula [1]:

$$\begin{aligned} c(z,T,S) = {} & 1448.96 + 4.591\,T - 5.304\times10^{-2}T^2 \\ & + 2.374\times10^{-4}T^3 + 1.340(S-35) + 1.630\times10^{2}z \\ & + 1.675\times10^{-7}z^2 - 1.025\times10^{-2}T(S-35) \\ & - 7.139\times10^{-13}Tz^3 \end{aligned} \tag{5.17}$$

In these equations the units of S should be given in parts per thousand, the values of temperature in Celsius degrees and the values of depth in metres. The calculation of partial derivatives allows to notice that, to a good approximation

$$\frac{\partial c}{\partial T} \approx 4.6 \text{ m/s/°C} \quad , \quad \frac{\partial c}{\partial S} \approx 1.34 \text{ m/s/ppt} \quad , \quad \frac{\partial c}{\partial z} \approx 0.016 \text{ m/s/m}$$

which can be translated in terms of variations as

- $\delta T = 25°\text{C} \Rightarrow \delta c = 115\,\text{m/s}$,
- $\delta S = 5$ ppt $\Rightarrow \delta c = 6.5\,\text{m/s}$,
- $\delta z = 6\,\text{km} \Rightarrow \delta c = 96\,\text{m/s}$,

that is, salinity is the parameter with the least impact in the variation of sound speed. There are other empirical expressions that combine different sets of parameters; however, one of the most important expressions, which depends on depth only, is the so-called *Munk's profile*, or deep water profile, which is given by [6]:

$$c(z) = c_1\left[1 + \epsilon\,(\eta + e^{\eta} - 1)\right] \tag{5.18}$$

where $\epsilon = 7.4 \times 10^{-3}$, $\eta = 2(z - z_1)/B$ and $B = 1.3\,\text{km}$; c_1 is a parameter of reference and z_1 corresponds to the depth where the minimum of $c(z)$ is located. The existence of this minimum has, as a consequence, the formation of a propagation channel known as the *SOFAR* channel (from *SOund Fixing And Ranging*).

5.7 Wave Reflection

When a wave is incident on a boundary separating different media, part of its energy is transferred from one medium to the other and part of its energy returns to the original medium; this phenomenon is known as wave *reflection*.

Let us consider the incidence, reflection and transmission of a plane wave at a boundary between two media with different densities and speeds of propagation (see Fig. 5.3). Without loss of generality we can write the incident, reflected and transmitted waves as

$$p_i = \exp\left[ik_1 \left(x \sin\theta_1 - z \cos\theta_1\right)\right]$$

$$p_r = \mathcal{R} \exp\left[ik_1 \left(x \sin\theta_1 + z \cos\theta_1\right)\right]$$

$$p_t = \mathcal{T} \exp\left[ik_2 \left(x \sin\theta_2 - z \cos\theta_2\right)\right]$$

where $k_{1,2} = \omega/c_{1,2}$, and with $\mathcal{R}$ and $\mathcal{T}$ representing, respectively, the *reflection coefficient* and the *transmission coefficient*[2]. At the boundary separating the two media ($z = 0$) the condition

$$p_i + p_r = p_t \tag{5.19}$$

implies that

$$1 + \mathcal{R} = \mathcal{T} \tag{5.20}$$

together with

$$k_1 \sin\theta_1 = k_2 \sin\theta_2$$

This identity is nothing more than Eq. (5.13) rewritten with wavenumbers. Let us notice now that Eq. (5.19), combined with Eq. (5.20), implies that $\mathcal{T} \to 0$ and $\mathcal{R} \to -1$ when $\theta_1 \to 90°$, meaning that wave transmission *decreases* with the angle of incidence θ, while reflection *increases*; for the grazing angle $\underline{\theta}$ the effect is the opposite one: wave reflection *decreases* when the grazing angle *increases*. Equation (5.19), together with Eq. (5.20), does not allow to take into account the media

[2] In some references these coefficients are defined in terms of wave *intensities* (see [20]).

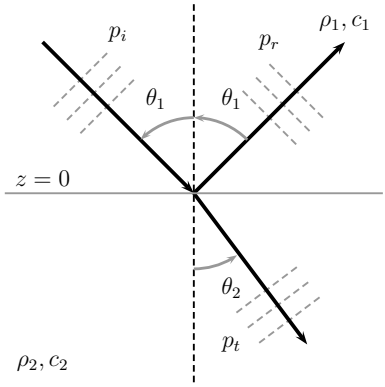


Fig. 5.3 Incidence, reflection and transmission of a plane wave at the boundary between two media with different properties

densities. The second condition that is required at the boundary $z = 0$ corresponds to the continuity of the vertical component of the particle velocity:

$$v_{z,1} = v_{z,2}$$

which can be calculated using the corresponding potential (see Eq. (5.9)); in the upper and lower media this potential corresponds, respectively, to

$$\Phi_1 = -\frac{i}{\rho_1 \omega} \left\{ \begin{array}{r} \exp\left[ik_1 \left(x \sin\theta_1 - z \sin\theta_1\right)\right] \\ + \mathcal{R} \exp\left[ik_1 \left(x \sin\theta_1 + z \sin\theta_1\right)\right] \end{array} \right\}$$

and

$$\Phi_2 = -\frac{i}{\rho_2 \omega} \mathcal{T} \exp\left[ik_2 \left(x \sin\theta_2 - z \sin\theta_2\right)\right]$$

After the calculation of derivatives, and by the continuity of v_z, we can obtain the expression

$$1 - \mathcal{R} = \mathcal{T} \frac{\rho_1 k_2 \sin\theta_2}{\rho_2 k_1 \sin\theta_1} \tag{5.21}$$

which combined with Eq. (5.20) allows to conclude what

$$\mathcal{T} = \frac{2\rho_2 k_1 \cos\theta_1}{\rho_1 k_2 \cos\theta_2 + \rho_2 k_1 \cos\theta_1} \tag{5.22}$$

and

$$\mathcal{R} = \frac{\rho_2 k_1 \cos\theta_1 - \rho_1 k_2 \cos\theta_2}{\rho_1 k_2 \cos\theta_2 + \rho_2 k_1 \cos\theta_1} \tag{5.23}$$

The corresponding expressions in terms of velocities correspond to [3]

$$\mathcal{T} = \frac{2\rho_2 c_2 \cos\theta_1}{\rho_2 c_2 \cos\theta_1 + \rho_1 c_1 \cos\theta_2} \tag{5.24}$$

and

$$\mathcal{R} = \frac{\rho_2 c_2 \cos\theta_1 - \rho_1 c_1 \cos\theta_2}{\rho_2 c_2 \cos\theta_1 + \rho_1 c_1 \cos\theta_2} \tag{5.25}$$

Let us consider some particular cases of Eq. (5.25):

- For a *normal* incidence (that is, when $\theta_1 = 0$) we can obtain that

$$\mathcal{R} = \frac{Z_2 - Z_1}{Z_2 + Z_1}$$

and

$$\mathcal{T} = \frac{2Z_2}{Z_2 + Z_1}$$

where $Z_1 = \rho_1 c_1$ and $Z_2 = \rho_2 c_2$ represent the media impedances. The condition

$$Z_1 = Z_2$$

is known as *impedance matching*.
- In the case when $\rho_1 = \rho_2$ and because of Eq. (5.13) we can obtain that $\mathcal{R} = 0$; this result allows to introduce an interpretation of wave reflection as a direct consequence of the difference in densities between the propagation media.
- In the case when $\rho_2 \ll \rho_1$ (which represents, approximately, the air–water boundary) we can obtain that $\mathcal{R} \approx -1$ and $\mathcal{T} \approx 0$; that means that a strong contrast of densities makes impossible for the wave to be transmitted from the old to the new medium.
- In the case when $\rho_2 \gg \rho_1$ (which represents, approximately, the boundary water–basalt) we can obtain that $\mathcal{R} \approx 1$ and $\mathcal{T} \approx 2$, which seems rather counter-intuitive in terms of amplitudes; this conclusion can be clarified noticing that in terms of intensities

$$\frac{I_t}{I_i} = \frac{p_t^2/(\rho_2 c_2)}{p_i^2/(\rho_1 c_1)} = \mathcal{T}^2 \, \frac{\rho_1 c_1}{\rho_2 c_2} \ll 1$$

In the general case the complexity of reflection increases due to the following factors:

- The wave that was propagating in the water is reflected on an elastic bottom; therefore, the incident wave generates in the bottom the propagation of a compressional and a shear wave (see Fig. 5.4).
- Wave propagation in an elastic bottom can exhibit a dissipative behaviour.

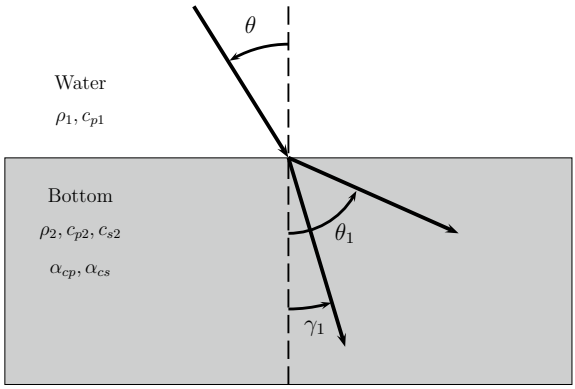


Fig. 5.4 Wave propagation from water to an elastic bottom

Under these conditions Eq. (5.25) can be replaced with [31]:

$$\mathcal{R}(\theta) = \frac{D(\theta)\cos\theta - 1}{D(\theta)\cos\theta + 1}, \tag{5.26}$$

where

$$D(\theta) = A_1\left(A_2\frac{1-A_7}{\sqrt{1-A_6^2}} + A_3\frac{A_7}{\sqrt{1-A_5/2}}\right),$$

$$A_1 = \frac{\rho_2}{\rho_1} \quad , \quad A_2 = \frac{\tilde{c}_{p2}}{c_{p1}} \quad , \quad A_3 = \frac{\tilde{c}_{s2}}{c_{p1}}$$

$$A_4 = A_3\sin\theta \quad , \quad A_5 = 2A_4^2$$

$$A_6 = A_2\sin\theta \quad , \quad A_7 = A_5(2-A_5)$$

$$\tilde{c}_{p2} = c_{p2}\frac{1-i\tilde{\alpha}_{cp}}{1+\tilde{\alpha}_{cp}^2} \quad , \quad \tilde{c}_{s2} = c_{s2}\frac{1-i\tilde{\alpha}_{cs}}{1+\tilde{\alpha}_{cs}^2} \quad ,$$

$$\tilde{\alpha}_{cp} = \frac{\alpha_{cp}}{40\pi\log e} \quad , \quad \tilde{\alpha}_{cs} = \frac{\alpha_{cs}}{40\pi\log e} \quad ,$$

where the units of attenuation of compressional and shear waves should be given in dB/λ. Often the reflection coefficient is real when $\alpha_{cp} = \alpha_{cs} = 0$, and the angle of incidence is inferior to the critical angle θ_{cr}, which is given by

$$\theta_{cr} = \arcsin\left(\frac{c_{p1}}{c_{p2}}\right) \tag{5.27}$$

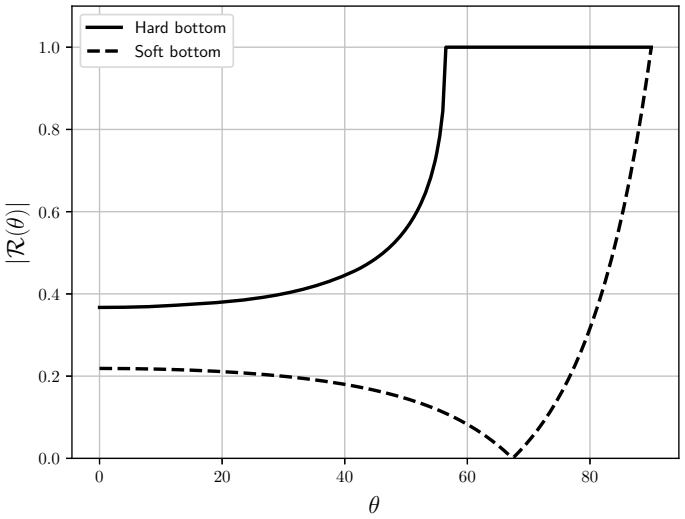


Fig. 5.5 Reflection coefficient as a function of the angle of incidence, for a hard and a soft bottom

This expression makes sense only when $c_{p1} < c_{p2}$. The transfer of energy to the elastic medium is of little significance when $\theta < \theta_{cr}$, and for small values of θ only a reduced part of the original energy allows the generation of shear waves in the bottom. Analytical expressions for systems with more than two boundaries are discussed in detail in [2] and [3].

The behaviour of the reflection coefficient for two different bottom types is illustrated in Fig. 5.5 (generated by the Python Code Snippet 5.3); as expected $|\mathcal{R}(\theta)| \to 1$ as the angle of incidence increases, but the progressive variation of the reflection coefficient is radically different from case to case: for a "hard" bottom ($c_{p1} < c_{p2}$) the reflectivity increases progressively, until it achieves the maximal value at $\theta_{cr} = \arcsin(1500/1800) = 56°$; for a "soft" bottom ($c_{p1} > c_{p2}$) the reflectivity decays progressively, until becoming 0 at 67°, and then raise again until achieving a maximal value at 90°. The properties of some bottom types are indicated in Table 5.1, with $\left[c_p\right] = [c_s]$ = m/s, $[\rho]$ = kg/m^3 and $\left[\alpha_p\right] = [\alpha_s] = $ dB/λ [3]; the relationship between attenuation units corresponds to

$$\text{dB/m} = \frac{\text{dB}/\lambda}{\lambda}$$

Table 5.1 Acoustic properties of some bottom types

Bottom	c_p (m/s)	c_s (m/s)	ρ (kg/m^3)	α_p (dB/λ)	α_s (dB/λ)
Mud	1600	400	1800	0.2	1.8
Gravel	1700	180	1500	0.8	1.5
Sand	1800	600	2000	0.1	2.0
Sandy clay	1820	260	1700	0.4	0.6
Coarse sand	1950	350	2100	0.5	0.9
Moraine	1950	600	2100	0.4	1.0
Chalk	2400	1000	2200	0.2	0.5
Calcarenite	2800	1400	2400	0.1	0.2
Limestone	3000	1500	2400	0.1	0.2
Basalt	5250	2500	2700	0.1	0.2

Code Snippet 5.3 Reflection coefficient for an elastic bottom.

```
from numpy import *
from matplotlib.pyplot import *

rcParams.update({"text.usetex": True})

def bdryr(rho1=None,rho2=None,
          cp1=None,cp2=None,
          cs2=None,alphap=None,
          alphas=None,theta=None):

        e = exp(1)

        tilap = alphap/( 40.0*pi*log10( e ) )
        tilas = alphas/( 40.0*pi*log10( e ) )

        tilcp2 = cp2*(1.0-1j*tilap)/\
                     (1.0+tilap*tilap)
        tilcs2 = cs2*(1.0-1j*tilas)/\
                     (1.0+tilas*tilas)

        a1 = rho2/rho1
        a2 = tilcp2/cp1
        a3 = tilcs2/cp1
        a4 = a3*sin( theta )
        a5 = 2.0*a4*a4
        a6 = a2*sin( theta )
        a7 = 2.0*a5 - a5*a5

        d = a1*( a2*(1.0-a7)/ \
            sqrt(1.0-a6*a6)+a3*a7/sqrt(1.0-0.5*a5) )

        R = (d*cos(theta)-1.0)/(d*cos(theta)+1.0)

        return R
```

```
theta = linspace(0,90,181)
thetar = theta*pi/180

# Soft bottom:
cw = 1500.0
cp = 1300.0
cs = 0.0
alphap = 0.0
alphas = 0.0
rhow = 1000.0
rhob = 1800.0

R = bdryr(rhow,rhob,
cw,cp,cs,alphap,alphas,thetar)

Rs = abs( R )

# Hard bottom:
cw = 1500.0
cp = 1800.0
cs = 0.0
alphap = 0.0
alphas = 0.0
rhow = 1000.0
rhob = 1800.0

thetacr = 180.0/pi*arcsin( cw/cp )
print( thetacr )

R = bdryr(rhow,rhob,
cw,cp,cs,alphap,alphas,thetar)

Rr = abs( R )

figure(1)
plot(theta,Rr,'k'  ,linewidth=2,
label='Hard bottom')
plot(theta,Rs,'k--',linewidth=2,
label='Soft bottom')
xlabel(r'$\theta$',fontsize=18)
ylabel(r'$|{\cal R}(\theta)|$',fontsize=18)
ylim(0,1.1)
legend(loc=2)
grid(True)

show()
```

5.8 Beam Displacement

In the discussion of the reflection coefficient we considered the incidence of a plane wave on the boundary that separates two different media. Let us consider now the incidence of a *wavebeam*, which at the boundary can be represented as

$$p_i = \int_{\underline{k}_{min}}^{\underline{k}_{max}} A(\underline{k})\; e^{i\underline{k}x}\; d\underline{k} \tag{5.28}$$

where $\underline{k} = (\omega/c)\sin\theta$ represents the horizontal component of the wavenumber and $A(\underline{k})$ represents the amplitude of each plane wave. For the reflected beam we can write that

$$p_r = \int_{\underline{k}_{min}}^{\underline{k}_{max}} A(\underline{k})\; \mathcal{R}\left(\underline{k}\right)\; e^{i\underline{k}x}\; d\underline{k}$$

where $\mathcal{R}$ represents the reflection coefficient. Since the main goal of this section is to consider phase corrections to the exponential let us rewrite the reflection coefficient in the form

$$\mathcal{R} = |\mathcal{R}|\, e^{i\phi(\underline{k})}$$

where ϕ represents the phase of $\mathcal{R}$; therefore, the expression for the reflected beam becomes

$$p_r = \int_{\underline{k}_{min}}^{\underline{k}_{max}} A(\underline{k})\;\; |\mathcal{R}|\;\; e^{i\left(\underline{k}x+\phi\right)}\; d\underline{k}$$

The phase has a smooth behaviour except near the critical angle, which allows to introduce the approximation

$$\phi(\underline{k}) \approx \phi(\underline{k}_0) + \left(\underline{k} - \underline{k}_0\right)\left(\frac{d\phi}{d\underline{k}}\right)$$

After substitution into the expression for the reflected beam one can obtain that

$$p_r = \mathcal{R}\left(\underline{k}_0\right)\; e^{-ik_0\left(d\phi/d\underline{k}\right)} \int_{\underline{k}_{min}}^{\underline{k}_{max}} A(\underline{k})\;\; e^{i\left(\underline{k}x-\Delta\right)}\; d\underline{k} \tag{5.29}$$

where

$$\Delta = -\frac{d\phi}{d\underline{k}} \tag{5.30}$$

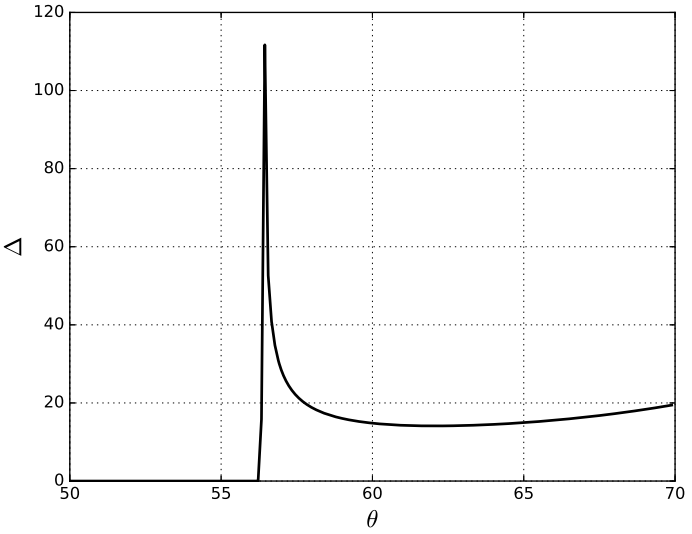


Fig. 5.6 Beam displacement near the critical angle

stands for the *beam displacement* [2, 32]. This denomination becomes clear when we compare Eq. (5.29) with Eq. (5.28) and notice that the reflected beam includes not only an amplitude correction to the incident beam, but also a *geometric* correction; in terms of incidence of a single ray this correction implies, surprisingly, that the ray should propagate along the boundary along a distance Δ before "returning" to the water column, following an specular direction. The dependence of beam displacement on frequency can be clarified rewriting Eq. (5.30) in order for the derivative to depend on the angle of incidence:

$$\Delta = -\frac{1}{\cos\theta}\left(\frac{c}{\omega}\right)\frac{d\phi}{d\theta} \tag{5.31}$$

Obviously, one can obtain that $\Delta \to 0$ when $\omega \to \infty$. Figure 5.6, generated by the Python Code Snippet 5.4, illustrates the calculation of Δ in the proximity of the critical angle for the hard bottom previously considered.

Code Snippet 5.4 Beam displacement calculation.

```
from numpy import *
from matplotlib.pyplot import *

rcParams.update({"text.usetex": True})

def bdryr(rho1=None,rho2=None,
          cp1=None,cp2=None,
```

```
            cs2=None,alphap=None,
            alphas=None,theta=None):

        e = exp(1)

        tilap = alphap/( 40.0*pi*log10( e ) )
        tilas = alphas/( 40.0*pi*log10( e ) )

        tilcp2 = cp2*(1.0-1j*tilap)/\
                     (1.0+tilap*tilap)
        tilcs2 = cs2*(1.0-1j*tilas)/\
                     (1.0+tilas*tilas)

        a1 = rho2/rho1
        a2 = tilcp2/cp1
        a3 = tilcs2/cp1
        a4 = a3*sin( theta )
        a5 = 2.0*a4*a4
        a6 = a2*sin( theta )
        a7 = 2.0*a5-a5*a5

        d = a1*( a2*(1.0-a7)/\
            sqrt(1.0-a6*a6)+a3*a7/sqrt(1.0-0.5*a5) )

        R = (d*cos(theta)-1.0)/(d*cos(theta)+1.0)

        return R

# Grazing angle:
theta = linspace(50,70,181)
thetar = theta*pi/180
dtheta = diff( thetar )

f = 150.0
w = 2*pi*f

# Hard bottom:
cw = 1500.0
cp = 1800.0
cs = 0.0
alphap = 0.0
alphas = 0.0
rhow = 1000.0
rhob = 1800.0
k = w/cw
kh = k*sin( thetar )

R = bdryr(rhow,rhob,
cw,cp,cs,alphap,alphas,thetar)

rR = real( R )
iR = imag( R )
phi = arctan2( iR, rR )
dphi = diff( phi )
dk = diff( kh )
displacement = -dphi/dk

figure(1)
```

```
plot(theta[0:-1],displacement,'k',linewidth=2)
xlabel(r'$\theta$',fontsize=18)
ylabel(r'$\Delta$',fontsize=18)
grid(True)

show()
```

5.9 Final Remarks

In the following two chapters we will deviate from the formalism adopted so far: Chapter 6 will introduce an "operational" perspective of Underwater Acoustics, based on the SONAR equation, which will be followed in Chap. 7 by the discussion of transducers arranged into array systems. Before we proceed, and in order to consolidate our current knowledge, it is important to enumerate the topics discussed so far, which corresponded to:

- Propagation of acoustic waves in a boundless medium, with constant velocity.
- Plane, cylindrical and spherical waves.
- Phase velocity and group velocity.
- Wave attenuation in the ocean.
- The Doppler effect.
- Propagation of acoustic waves in an elastic medium.
- Different types of wave equation, and relationships between particle displacement, particle velocity and acoustic pressure as well as their association with the corresponding potentials.
- The definition of acoustic impedance.
- Wave dispersion.
- Wave refraction.
- Wave reflection.
- Beam displacement.

The relevance of these topics will become evident when we come back to the discussion of the wave equation and wave propagation, this time in the context of the propagation models, to take place in Chap. 8.

Chapter 6
The SONAR Equation

6.1 Introduction

The *SONAR* (abbreviation for *SOund NAvigation and Ranging*) equation quantifies the performance of a detection system in Underwater Acoustics. The equation corresponds, in a certain sense, to a definition of the signal-to-noise ratio, which considers three fundamental components of a typical detection system: the acoustic source, the propagation medium and the reception system. Such definition of the signal-to-noise ratio can be written in general terms as [33]

$$\text{SNR} = \frac{\text{Signal}}{\text{Transmission Loss}} \times \frac{\text{Array Gain}}{\text{Total noise}} \tag{6.1}$$

Since the "original" SNR is calculated in dB it should not be a surprise that the SONAR equation corresponds to an algebraic sum of terms with different signs, each given in dB. In fact, there is no one, but two SONAR equations: the *passive* SONAR equation and the *active* SONAR equation. We have been delaying their discussion in order to properly introduce the concept of the SNR in Chap. 3; such discussion allowed us to introduce already an important term of the SONAR equation, namely, the *Detection Threshold* or DT (see Sect. 3.7). As we shall see the issues of wave propagation accounting or not for the existence of boundaries, presented in Chaps. 4 and 5, will allow to provide a solid basis for the introduction of another term of the equation. The list that follows indicates the typical denomination given to the different terms [1, 20]:

- SE or Signal Excess;
- SL or Source Level;
- TL or Transmission Loss;
- NL or Noise Level;
- DI or Directivity Index;
- TS or Target Strength;
- RL or Reverberation Level.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023

O. C. Rodríguez, *Fundamentals of Underwater Acoustics*,
https://doi.org/10.1007/978-3-031-31319-6_6

Before we describe each term it is important to illustrate the manner in which the different terms are related to each other in the SONAR equations.

6.2 The Passive SONAR Equation

In the operation of a passive SONAR the detection system simply records ambient noise, looking to detect noise variations that could indicate the presence of a signal emitted by an acoustic source; under such conditions the performance of the detection system is quantified by the *passive sonar equation*, which is written as

$$\mathrm{SE} = (\mathrm{SL} - \mathrm{TL}) - (\mathrm{NL} - \mathrm{DI}) - \mathrm{DT} \tag{6.2}$$

6.3 The Active SONAR Equation

In the operation of an active SONAR system the detection system transmits a signal periodically; in the presence of a given *target* the signal is reflected and detected as an echo. If the reception system coincides with the transmission system the SONAR configuration is called *monostatic*; depending on the multiplicity of the transmission and/or reception systems the SONAR configuration is called *bistatic* or *multistatic*. In the operation of an active SONAR system there are two factors that use to be differentiated: the ambient *reverberation* (that is, additional echos generated by the boundaries and many objects in the water) and the ambient noise. The differentiation of these factors is represented in the *active sonar equation* as

$$\mathrm{SE} = \mathrm{SL} - 2\mathrm{TL} + \mathrm{TS} - X + \mathrm{DI} - \mathrm{DT} \tag{6.3}$$

where

$$X = \begin{cases} \mathrm{RL} \text{ if reverberation is the dominant factor,} \\ \mathrm{NL} \text{ if ambient noise is the dominant factor.} \end{cases}$$

The doubling of the transmission loss in the equation represents the fact that the signal propagates from the source to the target and backwards, but there are variants of Eq. (6.3) which can account for different transmission losses at transmission and reception.

6.4 The Signal Excess

The SE characterizes the intensity of the signal which rises over the intensity of the "total" noise; detection will be successful when $\mathrm{SE} > 0$ and will fail if $\mathrm{SE} < 0$. In the case when $\mathrm{SE} = 0$ the SONAR equation can be rewritten setting the transmission

loss equal to the other terms of the equation, which defines the *Figure of Merit* (abbreviated as *FOM*). In this way the FOM corresponds to the value of transmission loss required to have a signal excess of zero dB. For instance, in the case of Eq. (6.2) one would have that

$$\text{FOM} = \text{SL} - (\text{NL} - \text{DI} + \text{DT}) \tag{6.4}$$

For Eq. (6.3) and in the case of ambient noise the equation corresponds to

$$\text{FOM} = 2\text{TL} = \text{SL} + \text{TS} - \text{NL} + \text{DI} - \text{DT} \tag{6.5}$$

However, when the reverberation is the dominant factor the FOM that follows from Eq. (6.3) is not representative of the performance of the SONAR system [1].

6.5 The Source Sound Level

The SL represents the relationship between the intensity $I_s(r)$ of the source and the intensity of reference for a point source I_{ref}:

$$\text{SL} = 10 \log \frac{I_s(r = 1\text{ m})}{I_{ref}} \tag{6.6}$$

For the calculation of I_{ref} the following parameters are considered:

$$p_{ref} = 1\,\mu\text{Pa},\ \rho = 1000\,\text{kg/m}^3,\ c = 1500\,\text{m/s},$$

producing the value $I_{ref} = p_{ref}^2/(\rho c) = 6.67 \times 10^{-19}$ W/m^2. For practical effects the SL is calculated resorting to the source power P_* in watt; in this case [20]

$$I_s(r = 1\text{ m}) = P_*/(4\pi r^2) = P_*/(4\pi\,(1)^2) = P_*/(4\pi)$$

which allows to rewrite Eq. (6.6) as

$$\text{SL} = 10 \log\ P_* + 170.77\text{ dB} \tag{6.7}$$

6.6 The Transmission Loss

The transmission loss represents the relationship between the intensity of reference and the source intensity at any given location:

$$\text{TL} = 10 \log \frac{I_{ref}}{I} \tag{6.8}$$

The difference between the SL and the TL corresponds to the *Sound Pressure Level*, abbreviated as SPL:

$$\text{SPL} = \text{SL} - \text{TL}$$

Taking into account that $I = p^2/(\rho c)$ it is possible to *approximate* the calculation of TL from acoustic pressure as

$$\text{TL} = 20 \log \left| \frac{p_{ref}}{p} \right| \tag{6.9}$$

In the particular case of a cylindrical or a spherical wave, and including attenuation, we can write the pressure field as

$$p(r) = p_{ref}\ r^{-n} 10^{-\alpha r/20}$$

where $n = 1$ for a spherical wave and $n = 1/2$ for a cylindrical wave. Substituting into Eq. (6.9) we can obtain that

$$\text{TL} = \text{TL}_g + \text{TL}_d$$

where the *geometric* transmission loss corresponds to

$$\text{TL}_g = 20n \log r$$

and the *dissipative* transmission loss corresponds to

$$\text{TL}_d = \alpha r$$

In these expressions the units of r and α correspond, respectively, to m and dB/m. For *spherical spreading* $n = 1$ and the transmission loss becomes

$$\text{TL}_g = 20 \log r$$

while for *cylindrical spreading* $n = 1/2$ and

$$\text{TL}_g = 10 \log r$$

In the general case an accurate prediction of TL requires resorting to a *propagation model*, which is the topic of discussion of Chap. 8; for model purposes the reference pressure is irrelevant because it can be always chosen as the initial field amplitude.

6.7 The Noise Level

The NL represents the relationship between the noise intensity I_N in general and the intensity of reference:

$$\mathrm{NL} = 10 \log \frac{I_N}{I_{ref}} \tag{6.10}$$

Noise itself is constituted by *self*-noise and by *ambient* noise. Self-noise is specific to the equipment that supports the reception system. Characterization of ambient noise is a problem of great complexity in Underwater Acoustics because of the numerous sources of noise, such as tides, seismic activity, sea state due to the wind on its surface and marine traffic; in this context, the different marine species that communicate by acoustic means can be considered also as sources of noise. All these factors combined create a complex map of noise intensity, which is frequency dependent and it is represented by *Wenz curves* [34, 35].

As a concluding remark of this section, it is worth noticing that the map of ambient noise from an empirical perspective (certainly open to discussion) can be grouped into the following bands/sources:

- $f < 1$ Hz: noise generated by tides and submarine earthquakes.
- $1 < f < 20$ Hz: noise generated by internal tides and upwelling.
- $20 < f < 500$ Hz: noise generated by surface ships, essentially engine noise and propeller cavitation noise; it is local dependent; for instance, ship traffic is considered to be "heavier" in the Mediterranean than in the Pacific. $500\,\mathrm{Hz} < f < 50$ KHz: noise generated by surface undulation, which depends on the *sea state*, defined by the *Beaufort scale* [20].
- $f > 50$ kHz: noise created by thermal sources.

6.8 The Directivity Index

The DI, which often coincides with the *Array Gain* (abbreviated as AG) [28], characterizes the directional gain of an array of transducers, which is the result of the specific geometry of its elements. For instance, for an array with a rectangular shape of sides L_x and L_y, the DI corresponds to

$$\mathrm{DI} = 10 \log \left(4\pi \frac{L_x L_y}{\lambda^2} \right) \tag{6.11}$$

In the general case the DI can be defined as

$$\mathrm{DI} = 10 \log D \tag{6.12}$$

where D represents the *directivity*, which can be written as [20]

$$D = I_B^{-1} \tag{6.13}$$

where

$$I_B = \int_0^{\pi/2} B(\theta)\ \cos\theta\ d\theta \tag{6.14}$$

In this expression $B(\theta)$ represents the array *beam pattern*, which corresponds to the ratio between the mean transmitted/received array power for a particular direction θ, and the mean transmitted/received array power for normal incidence ($\theta = 0$) [20]; the detailed discussion of the beam pattern will take place in Chap. 7.

6.9 The Target Strength

The TS represents the relationship between the intensity of the incident wave at a target I_i and the intensity of the reflected wave I_r:

$$\text{TS} = 10\log\frac{I_r}{I_i} = 10\log A \tag{6.15}$$

where A depends in general on the target's shape, the angle of incidence and the wavenumber. The following list presents some of the simplest cases [20]:

- Sphere of radius a, under the condition that $ka \ll 1$:

$$A = 61.7\ V^2/\lambda^4$$

  where $V = (4/3)\pi a^3$.
- Sphere of radius a, under the condition $ka \gg 1$:

$$A = a^2/4$$

- Surface with curvature radii a_1 and a_2, normal incidence, under the condition that $ka_1 \gg 1$ and $ka_2 \gg 1$:

$$A = a_1\ a_2/4$$

- Cylinder of radius a and length l, normal incidence to the cylinder axis, under the condition that $ka \gg 1$:

$$A = \frac{al^2}{2\lambda}$$

  For an angle of incidence θ relative to the cylinder's normal one would have that

$$A = \frac{al^2}{2\lambda}\left(\frac{\sin\beta}{\beta}\right)^2 \cos^2\theta$$

where $\beta = kl\sin\theta$

- Rectangular plate of sides a and b ($a > b$), incidence at an angle θ relative to the plate's normal, under the condition that $kb \gg 1$:

$$A = \left(\frac{ab}{\lambda}\right)^2 \left(\frac{\sin\beta}{\beta}\right)^2 \cos^2\theta$$

where $\beta = ka\sin\theta$

- Circular plate with radius a, perpendicular incidence to the plate, under the condition that $ka \ll 1$:

$$A = \left(\frac{4}{3\pi}\right)^2 k^4 a^6$$

- Circular plate of radius a, incidence with an angle θ relative to the plate's normal, under the condition that $ka \gg 1$:

$$A = \left(\frac{\pi a^2}{\lambda}\right)\left(\frac{2J_1(\beta)}{\beta}\right)^2 \cos^2\theta$$

where $\beta = 2ka\sin\theta$.

6.10 The Reverberation Level

The RL represents the relationship between the intensity of *reverberation* I_R and the intensity of reference:

$$\mathrm{RL} = 10\log\frac{I_R}{I_{ref}} \tag{6.16}$$

Reverberation is a complex phenomenon that can be considered as an extension of reflection, in which the propagation of the reflected signal can take into account other paths besides the one defined by specular reflection; this is possible because of the roughness of the reverberating surface, which at a microscopic level allows the return of energy *mostly* along the specular direction, but it is not confined to such direction *within and outside* of the plane of incidence. The phenomenon of reverberation is important in diverse areas of Underwater Acoustics; for instance, without bottom reverberation it would be impossible to create bathymetric charts with a sidescan SONAR. Given the complexity of the subject it will be addressed in more detail in Chap. 9.

6.11 Examples

Let us exemplify the application of the SONAR equation.[1]

1. *Passive* SONAR equation: an AUV transmits an echolocation burst periodically, with a frequency of 1 kHz. It is desired to determine its operational range (that is, the maximal horizontal distance relative to the locating ship, for which SE = 0), knowing that SL = 80 dB, DI = 20 dB, DT = 0 dB and that the sea state is 2. It is considered that the geometric spreading of transmission loss is cylindrical. To determine the desired distance one can proceed as follows:
   - From the Wenz curves [35] we can see that NL $\approx$ 60 dB at $f = 1$ kHz and for a sea state 2.
   - Let us calculate the FOM with Eq. (6.4):

$$\text{FOM} = 80 - (60 - 20 + 0) = 40 \text{ dB}$$

   - According to the Thorp model (see Sect. 4.5) $\alpha = 0.069$ dB/km $= 6.9 \times 10^{-5}$ dB/m at $f = 1$ kHz; the operational range r, taking into account both the attenuation and the cylindrical spreading is the solution of the equation

$$10 \log r + \alpha r = \text{FOM}$$

     and corresponds to $r \approx 8.7$ km.

2. *Active* SONAR equation: In an area of intense traffic a frigate is searching for a buoy that was detached by accident from its original location. The parameters of the SONAR system correspond to SL = 185 dB, DT = -3 dB, NL = 106 dB, DI = 27 dB, TS = 67 dB and TL = 48 dB. It is desired to determine if the buoy can be detected with such parameters.
   - Let us calculate the FOM with Eq. (6.5):

$$\text{FOM} = 185 + 67 - 106 + 27 + 3 = 176 \text{ dB}$$

   - We can see that for the given set of parameters it is valid that

$$\text{FOM} > 2\text{TL}$$

     which implies that the signal excess should be positive and the detection of the buoy should be possible.

[1] These examples are adapted from [20, 28].

Chapter 7
Arrays, Transducers and Vector Sensors

7.1 Introduction

An array of transducers, in the role of a *projector* or as a system of *receivers*, never will be *omnidirectional*; even if each transducer in the array can be idealized as a point device, the array will always give some *preference* to particular directions of transmission or reception in detriment of the remaining ones. This array feature can be exploited to increase the signal-to-noise ratio, and can be quantified determining the array beam pattern $B(\theta)$, which was already mentioned in Sect. 6.8. For instance, the wave transmitted by an array of arbitrary shape can be modulated in the far field as

$$p(r,\theta) \approx B(\theta)\,\frac{p_0}{r}\,e^{-i(kr-\omega t)} \tag{7.1}$$

Let us deepen in this chapter the discussion regarding the calculation of $B(\theta)$ and D starting with the case of the dipolar array, that is, an array with two transducers. The extension of this case to other configurations will allow to introduce the results for other types of arrays, often mentioned in the literature of Underwater Acoustics. The discussion will consider the incidence of plane waves on the array, but the results are valid for both signal transmission and reception. We will also introduce a brief description of vector sensors.

7.2 Array with Two Transducers

Let us consider a plane wave arriving with an angle θ to an array with a pair of "point" transducers, separated by a distance d (see Fig. 7.1). Admitting, without loss of generality, that the upper transducer is located at the origin of the coordinate system, we can write that

$$p_1(\theta,t) = p_0\cos(\omega t) \quad \text{and} \quad p_2(\theta,t) = p_0\cos(-\delta+\omega t)$$

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023

O. C. Rodríguez, *Fundamentals of Underwater Acoustics*,
https://doi.org/10.1007/978-3-031-31319-6_7

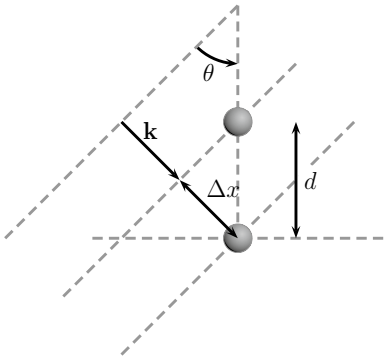


Fig. 7.1 Plane wave incidence on an array with two transducers

where p_1 and p_2 represent the arriving plane waves on the upper and lower transducers, respectively, and

$$\delta = k\Delta x = kd\sin\theta$$

represents the phase difference between the two waves. The array power $\mathcal{P}(\theta)$ is proportional to the mean value of $[p_1(\theta, t) + p_2(\theta, t)]^2$, thus we can write that [20]

$$B(\theta) = \frac{\mathcal{P}(\theta)}{\mathcal{P}(0)} = \frac{1 + \cos(kd\sin\theta)}{2} =$$

$$= \cos^2\left(\frac{kd\sin\theta}{2}\right) = \cos^2\left(\frac{\pi d\sin\theta}{\lambda}\right) \tag{7.2}$$

It follows from this expression that

$$I_B = \frac{1}{2}\left[1 + \frac{\sin(2\pi d/\lambda)}{2\pi d/\lambda}\right]$$

Let us see the calculation of the DI of this array for the set of parameters $d = 1\,\text{m}$, $f = 1000\,\text{Hz}$ and $c = 1500\,\text{m/s}$; we can obtain that $\lambda = 1.5\,\text{m}$ and $I_B = 0.397$, which corresponds to $\text{DI} = 10\log(1/0.397) = 4.02\,\text{dB}$.

The discussion of the beam pattern is by no means exhausted with the calculation of the DI. Let us consider the function $B(\theta)$ generated by Eq. (7.2) in the case when $d/\lambda = 2$, which is shown in Fig. 7.2; the position of the transducers is also illustrated in the figure to avoid any ambiguity. The angular selectivity angular of the array is revealed by a symmetric "butterfly" pattern, for which some specific directions for the value of $B(\theta)$ can be defined, namely:

- Directions of *maximal* power correspond to the angles $\theta_{\max}$ for which

$$B(\theta_{\max}) = 1$$

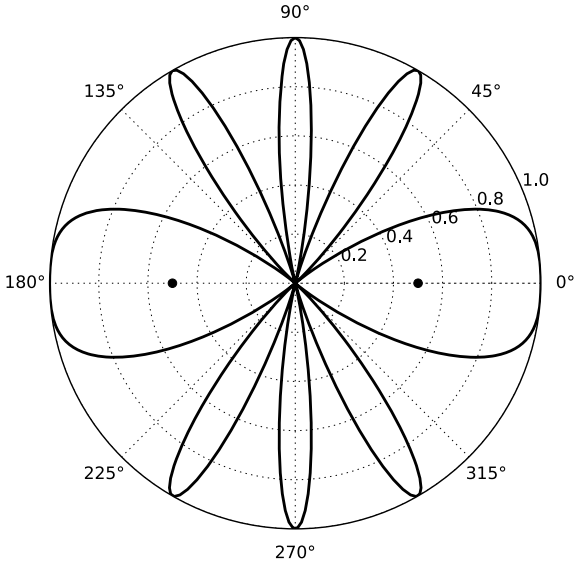


Fig. 7.2 Beam pattern of an array with two transducers aligned horizontally (pair of black dots), for $d/\lambda = 2$

For this specific array configuration such angles are given by the expression

$$\theta_{\max} = \sin^{-1}\left(\frac{n\lambda}{d}\right)$$

where $n = 0, 1, 2, 3 \ldots$

- Directions of *minimal* power correspond to the "nule" angles $\theta_{\min}$ for which

$$B(\theta_{\min}) = 0$$

which in this case are given by the expression

$$\theta_{\min} = \sin^{-1}\left(\frac{n\lambda}{2d}\right)$$

where $n = 1, 3, 5, \ldots$

- Directions of *average* power: correspond to the angles θ_{hp} for which

$$B(\theta_{hp}) = 1/2$$

which in this case are given by the expression

$$\theta_{hp} = \sin^{-1}\left(\frac{n\lambda}{4d}\right)$$

where $n = 1, 3, 5, \ldots$ The interval between two consecutive angles θ_{hp} defines a *beam width*; this is an important concept in terms of detection because it imposes a limit to the power of angular resolution of the array in the direction given by the centre of the interval.

For the directions perpendicular to the array axis, it is often considered that the beam pattern exhibits a symmetry of a figure of revolution.

7.3 Array with Several Transducers

The case of an array with N point transducers, linearly distributed at an interval given by a distance d, can be considered as an extension of the case of an array with two transducers; in such configuration, the calculation of the array power should take into account the progressive phase difference of each arriving wave relative to the first one, which allows to obtain that [20]

$$B(\theta) = \left[\frac{\sin(N\pi d \sin\theta/\lambda)}{N \sin(\pi d \sin\theta/\lambda)}\right]^2 \tag{7.3}$$

and

$$I_B = \frac{1}{N}\left\{1 + \frac{2}{N}\sum_{j=1}^{N-1}\left[\frac{(N-j)\sin(2\pi jd/\lambda)}{2\pi jd/\lambda}\right]\right\}$$

It should be noticed that expressions for the case of an array with two transducers should be obtained considering the case $N = 2$ in the expressions for $B(\theta)$ and I_B of this case. Additionally, the expressions already discussed for the angles $\theta_{\max}$ remain valid; however, it is necessary to deduce specific expressions for the other angles depending on the values of N and of d/λ.

As shown in Fig. 7.3, and in contrast with Fig. 7.1, the regular distribution of point transducers along a linear array allows to generate an extremely selective beam pattern, with a nule angular resolution along the array axis.

7.4 Linear Array

Idealizing the previous case as a continuous distribution of transducers along an array with length L it is possible to obtain that [20]

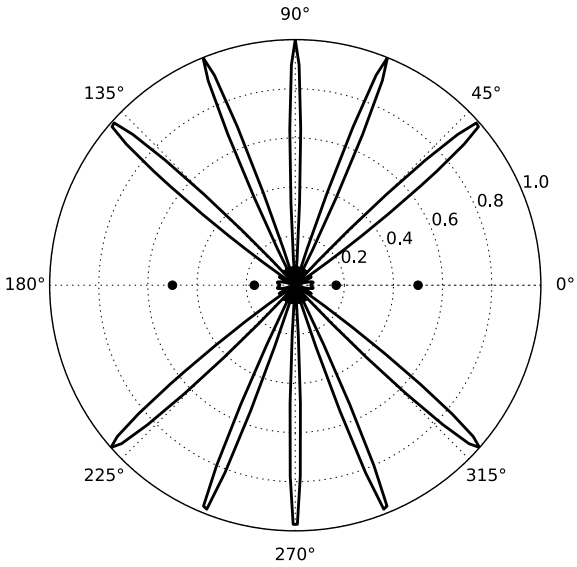


Fig. 7.3 Beam pattern for an array with four transducers aligned horizontally (black dots), for $d/\lambda = 8/3$

$$B(\theta) = \left[\frac{\sin\left(\frac{\pi L}{\lambda} \sin\theta\right)}{\frac{\pi L}{\lambda} \sin\theta} \right]^2 \tag{7.4}$$

and

$$\mathrm{DI} = 10 \log\left(\frac{2L}{\lambda}\right) \quad \text{which is valid for} \quad L \gg \lambda$$

We will stop at this point for a moment to introduce the concept of *array steering*: let us consider a modified version of Eq. (7.4) written as

$$B(\theta) = \left\{ \frac{\sin\left[\frac{kL}{2} \sin\left(\theta - \theta_0\right)\right]}{\frac{kL}{2} \sin\left(\theta - \theta_0\right)} \right\}^2 \tag{7.5}$$

If in Eq. (7.4) the maximal value of the beam pattern corresponded to the direction given by $\theta = 0$, then in Eq. (7.5) the maximal value corresponds now to the direction given by $\theta = \theta_0$; in other words, a linear array has the capability of "sweeping" a broad interval of directions, through the progressive change of the value of θ_0; this is illustrated by the Code Snippet 7.1. A frame of the animation for $\theta_0 = 45°$ is shown in Fig. 7.4.

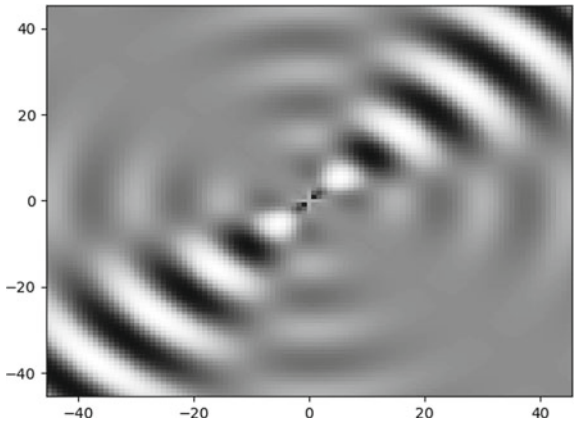


Fig. 7.4 $B(\theta)$ of a linear array on a Cartesian grid, for $\theta_0 = 45°$

Code Snippet 7.1 Array steering for a linear configuration of transducers

```
from numpy import *
from matplotlib.pyplot import *
from matplotlib import animation
from mpl_toolkits.mplot3d.axes3d import *

def data_gen(framenumber, p, plot):
    XB = 0.5*k*Larray*sin( THETA-theta0[framenumber] )
    p = sinc( XB/pi )*real( o )
    plot = pcolormesh(X, Y, p, shading='auto')
    return plot,

nframes  =  36
nperiods =   6
nx       = 101

freq  =  100.0; T = 1.0/freq; w = 2.0*pi*freq
c     = 1500.0
k     = w/c
# Wavelength:
L     = 2.0*pi/k
# Array length:
Larray = 2*L
tmin  =  0
tmax  =  T*nperiods
xmax  =  3.0*L
x = linspace(-xmax,xmax,nx)
y = linspace(-xmax,xmax,nx)

X,Y = meshgrid(x,y);
THETA = arctan2(Y,X)
R  = sqrt( X*X + Y*Y )
o = exp( 1j*k*R ) # Circular wave

theta0 = linspace(0,pi,nframes)
```

```
XB = 0.5*k*Larray*sin( THETA-theta0[0] )

# Define initial parameters:
p = sinc( XB/pi )*real( o )
fig = figure(1)
ax = fig.gca
plot = pcolormesh(X, Y, p, shading='auto')

# Animation:
animation = animation.FuncAnimation(fig,
data_gen, fargs=(p, plot),
frames=nframes,interval=1, blit=False)

show()
```

7.5 Disk Array

The rotation of a linear array along its centre allows to idealize an array with a disk shape of diameter D, for which [20]

$$B(\theta) = \left[\frac{2J_1\left(\frac{\pi D \sin\theta}{\lambda}\right)}{\frac{\pi D \sin\theta}{\lambda}} \right]^2 \tag{7.6}$$

and

$$\mathrm{DI} = 10\log\left(\frac{\pi D}{\lambda}\right)^2 \quad \text{which is valid for} \quad D \gg \lambda$$

where $J_1(x)$ represents the Bessel function of first order. In contrast with the previous cases the shape of the array implies automatically that the beam pattern should be symmetrical along the disk axis.

7.6 Frequency Response

In the previous sections, we have been characterizing the arrays of transducers in terms of their angular selectivity with mean power values calculated over time, and thus erasing any information related to the frequency. However, a transducer in the role of a projector is not going to exhibit a constant signal power over a given frequency interval, because its physical dimensions and properties imply that it can be better excited at some frequencies than at others, with the power increasing substan-

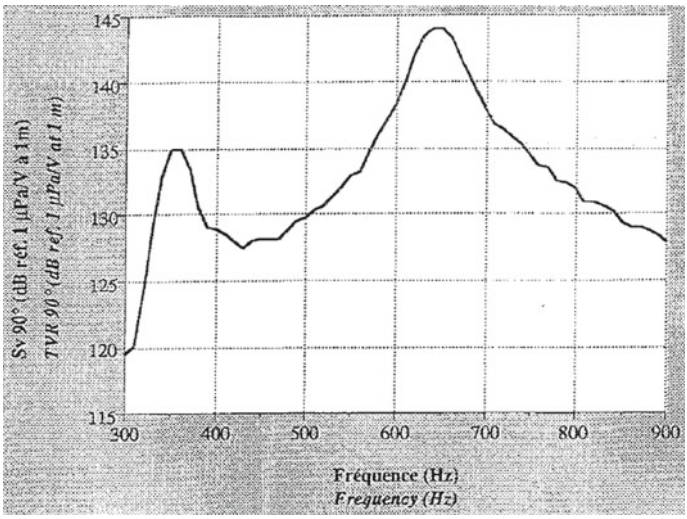


Fig. 7.5 Source spectrum of the ERAMER JH 360 source (courtesy of SiPLAB)

tially at the *resonant frequencies*; the dependence of the array power on frequency is known as the *frequency response* of the projector, and the curve that illustrates such dependence is called the *source spectrum*. For obvious reasons, the transmission of a signal in the proximity of a resonant frequency has advantages from an experimental point of view. Each source has a specific spectrum, which is determined experimentally, but all the spectra exhibit resonant frequencies. This is illustrated in Fig. 7.5, which shows the frequency response of the long-range acoustic source ERAMER JH 360 that exhibits two resonant frequencies, the first in the proximity 350 Hz and the second, higher, in the proximity 650 Hz.

7.7 Vector Sensors

A *Vector Sensor* is a device that allows to record the acoustic pressure and the particle velocity. In contrast with a typical transducer, a vector sensor allows to diversify the acquisition of acoustic data, through the registration of a scalar and a vector quantity, which are related to each other (see Chap. 5). This characteristic of a vector sensor can be exploited in different ways, in particular, to design "shorter" arrays relative to the ones designed with conventional transducers or to reduce the array's weight. In order to do so it is fundamental to develop a detailed characterization of the vector sensor in terms of its beam pattern, this time for both the scalar field of acoustic pressure and for the vector field of particle velocity; from the side of algorithms, one needs to extend the mathematical formalism, which is designed to account only for the scalar field of pressure, in order to combine into a single model the pressure and

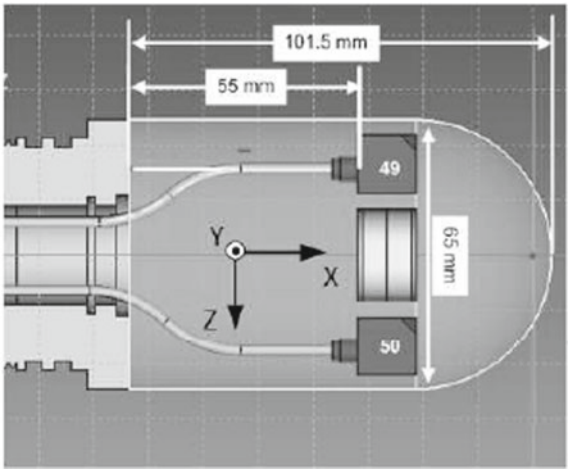


Fig. 7.6 DAVS schematics, showing the device dimensions, accelerometer pair and Cartesian coordinate system (courtesy of SiPLAB)

particle velocity fields. The technology is recent and it is still under development, but it has been used already for the characterization of the sea bottom [36].

Figure 7.6 shows a prototype of a vector sensor recently developed at SiPLAB, called *DAVS* (from *Dual Accelerometer Vector Sensor*), which combines a transducer with a pair of accelerometers aligned vertically. Being developed in Portugal makes the DAVS worth of some attention: it is a device with a cylindrical shape that can be used for the acquisition of acoustic signals in an underwater environment. The DAVS is constituted by two parts: the acoustic part, which contains two tri-axial accelerometers and a hydrophone moulded into a single unit and the container, which contains all the system chain of data acquisition, from the microcontroller to the battery compartment. The device has a length of 525 mm and a diameter of 65 mm; it is capable to acquire acoustic signals in autonomous operation thanks to its battery pack and a system of data storage; alternatively, it can operate in real time with an outer power source and data transfer via cable. Besides recording acoustic pressure the DAVS, obviously, can measure particle velocity in three dimensions, thanks to the pair of tri-axial accelerometers; the combination of pressure and particle velocity allows to obtain an extremely directional beam. On another side, thanks to its dual constitution, the DAVS allows the estimation of velocity gradients, a feature that can be exploited to increase the directionality of the beam's device. The DAVS is a compact device that can easily and rapidly be included in mobile platforms. Experimental tests with the DAVS mounted on an AUV of the Medusa class showed that such coupling does not compromise the manoeuvrability of the AUV; in the same tests, it was not detected any interference between the AUV motion and the acquisition of acoustic data [37]. The applications of the DAVS can include the detection and estimation of the direction of arrival of acoustic signals, source localization, target tracking, characterization of particle motion generated by benthonic invertebrate communities [38] and also for the characterization of sediments and acoustic communications [39, 40].

Chapter 8
Propagation Models

8.1 Introduction

In Underwater Acoustics, a *propagation model* is, essentially, a *computer model* that allows to calculate the pressure field, either by solving Eq. (4.2) or (in some cases) Eq. (4.1), for a given set of transmitted signals, speed of propagation, bottom and surface types, etc. In the context of the SONAR equation, the role of a propagation model is to provide accurate predictions of transmission loss, but the range of applications for model usage extends well beyond such perspective. The list of propagation models named in the literature is extense [5] and many, like, for instance, TRACEO [41, 42], TRACEO3D [43–45], Bellhop, KRAKEN, SPARC (models from the *Acoustic Toolbox* [29]), and the models integrated in SAGA (SNAP, PROSIM, ORCA, RAMGEO, etc.)[1] [46], or OASES [47], can be found online, and are distributed with an extense and detailed documentation, reference examples, and auxiliary code to create the model input and to visualize the results. This abundancy of resources is however of little benefit if the potential users lack a fundamental perspective of the differences between the models, of their specificities, advantages, and limitations. The fundamental goal of this chapter is to provide a fundamental basis for the general description of the different propagation models, and this is expected to be achieved by going through different aspects related to each model in a very specific way. The starting point for the discussion that follows is to group (in a very subjective manner) the models into four big "families", namely,

- Ray tracing models.
- Normal mode models.
- Fast-field models.
- Parabolic equation models.

[1] SAGA is an acoustic inversion software that allows to exploit Genetic Algorithms; it integrates several propagation models that can be used independently from inversion for the prediction of the acoustic field.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023

O. C. Rodríguez, *Fundamentals of Underwater Acoustics*,
https://doi.org/10.1007/978-3-031-31319-6_8

We will leave aside the discussion of propagation models based on a complete discretization of the waveguide, because they are used not as often as the previous ones and normally require significant computational resources. As in the previous chapter, the discussion will be complemented with Python Code Snippet.

8.2 Ray Tracing

The starting point for the discussion of ray tracing models is given by the following plane wave approximation as a solution to Eq. (4.2) [6]:

$$P(\omega) = Ae^{-i\omega\tau}\ , \tag{8.1}$$

where A represents an amplitude that changes slowly in comparison with the phase $\omega\tau$; the surfaces with τ = constant are known as *temporal fronts*. After substitution into the Helmholtz equation, together with a high-frequency approximation

$$k^2 \gg \frac{\nabla^2 A}{A}$$

we can obtain the pair of equations

$$\nabla\tau = \frac{1}{c} \tag{8.2}$$

known as the *Eikonal equation*, and

$$2\left(\nabla A \cdot \nabla\tau\right) + A\nabla^2\tau = 0 \tag{8.3}$$

known as the *transport equation*. Ray tracing corresponds to the calculation of trajectories through the integration of the Eikonal equation.

8.2.1 Ray Trajectories

The Eikonal equation can be further simplified by writing that

$$\frac{d\tau}{ds} = \frac{1}{c} \tag{8.4}$$

where ds represents the distance travelled by the acoustic wave. It follows from this simplification that

$$d\tau = \frac{ds}{c} \tag{8.5}$$

stands as the propagation time (or *travel* time) along the distance ds. The total propagation time between two points A and B corresponds then to

$$\tau = \int_A^B \frac{ds}{c} \tag{8.6}$$

where the trajectory that connects the two points should minimize the propagation time; such trajectories represent the propagation *rays*.[2] The trajectories that satisfy the condition of minimal time should obey to the following set of differential equation [41]:

$$\frac{d}{ds}\left(\frac{1}{c}\frac{dx}{ds}\right) = \frac{\partial}{\partial x}\left(\frac{1}{c}\right), \frac{d}{ds}\left(\frac{1}{c}\frac{dy}{ds}\right) = \frac{\partial}{\partial y}\left(\frac{1}{c}\right),$$
$$\frac{d}{ds}\left(\frac{1}{c}\frac{dz}{ds}\right) = \frac{\partial}{\partial z}\left(\frac{1}{c}\right) \tag{8.7}$$

We can see that the inverse of the speed of propagation appears constantly in the system of equations, and we can rely on this to define the *sound slowness* σ as

$$\sigma = \frac{1}{c} \tag{8.8}$$

which allows to rewrite the system given by Eq. (8.7) as

$$\frac{d}{ds}\left(\sigma\frac{dx}{ds}\right) = \frac{\partial\sigma}{\partial x}, \frac{d}{ds}\left(\sigma\frac{dy}{ds}\right) = \frac{\partial\sigma}{\partial y}, \frac{d}{ds}\left(\sigma\frac{dz}{ds}\right) = \frac{\partial\sigma}{\partial z} \tag{8.9}$$

The system can be written in a more compact way considering σ as a vector:

$$\boldsymbol{\sigma} = \left[\sigma_x, \sigma_y, \sigma_z\right] \tag{8.10}$$

in which

$$\frac{dx}{ds} = \frac{\sigma_x}{\sigma}, \frac{dy}{ds} = \frac{\sigma_y}{\sigma}, \frac{dz}{ds} = \frac{\sigma_z}{\sigma} \tag{8.11}$$

under these conditions Eq. (8.9) becomes

$$\frac{d\sigma_x}{ds} = \frac{\partial\sigma}{\partial x}, \frac{d\sigma_y}{ds} = \frac{\partial\sigma}{\partial y}, \frac{d\sigma_z}{ds} = \frac{\partial\sigma}{\partial z} \tag{8.12}$$

or in vector form:

$$\frac{d\boldsymbol{\sigma}}{ds} = \nabla\sigma \tag{8.13}$$

[2] In Tensor Calculus, the trajectories are called *geodesics*.

In a system with cylindrical symmetry, the Eikonal equation can be obtained simply by replacing x with r and by ignoring the derivatives in y, allowing to obtain that

$$\frac{dr}{ds} = \frac{\sigma_r}{\sigma}, \frac{dz}{ds} = \frac{\sigma_z}{\sigma}, \frac{d\sigma_r}{ds} = \frac{\partial \sigma}{\partial r}, \frac{d\sigma_z}{ds} = \frac{\partial \sigma}{\partial z} \tag{8.14}$$

with

$$\boldsymbol{\sigma}(s) = \left[\sigma_r(s), \sigma_z(s)\right]$$

The differential of the propagation time in these conditions corresponds to

$$d\tau = \frac{ds}{c} = \sigma \, ds$$

The "classical" form of the equations corresponds to [3]

$$\begin{aligned} \frac{dr}{ds} &= c(s)\sigma_r(s), \ \frac{d\sigma_r}{ds} = -\frac{1}{c^2}\frac{\partial c}{\partial r}, \\ \frac{dz}{ds} &= c(s)\sigma_z(s), \ \frac{d\sigma_z}{ds} = -\frac{1}{c^2}\frac{\partial c}{\partial z} \end{aligned} \tag{8.15}$$

or in vector form:

$$\frac{d\mathbf{r}}{ds} = c(s)\boldsymbol{\sigma}(s), \ \frac{d\boldsymbol{\sigma}}{ds} = -\frac{1}{c^2}\nabla c \tag{8.16}$$

When the speed of sound is a function only of depth we can obtain that

$$\frac{d\sigma_r}{ds} = 0$$

which leads us again to Snell's law along the propagation ray:

$$\sigma_r(s) = \frac{\cos \underline{\theta}(s)}{c(s)} = \text{ constant} \tag{8.17}$$

The points $(\hat{r}, \hat{z})$ in which $\underline{\theta} = 0$ are called *turning points*. The calculation of a particular trajectory requires the definition of the following initial conditions:

$$r(0) = r_0 \, , \quad z(0) = z_0 \, , \quad \sigma_r(0) = \frac{\cos \underline{\theta}(0)}{c(0)} \, , \quad \sigma_z(0) = \frac{\sin \underline{\theta}(0)}{c(0)}$$

where $\underline{\theta}(0)$ represents the *launching angle*, (r_0, z_0) represents the source's position, and $c(0)$ is the velocity of propagation at the source's position.

Boundary Reflections

The calculation of ray trajectories is not limited to the integration of the Eikonal; an additional element of the calculations is to account for boundary reflections, which requires stopping the process of integration. In order to restart the calculation of a given trajectory any ray tracing code needs to address the following issues:

- Determine the incidence point at the boundary; this point defines a new initial position.
- Apply the law of specular reflection to determine a new launching angle.

The integration is then restarted with the new position and the new launching angle. In the case of a flat boundary, the application of the law of specular reflection is trivial; in the general case, however, the determination of the launching angle can become cumbersome due to the ambiguity of trignometric functions. This problem can be overcome by resorting to Vector Calculus in the following way: let $\mathbf{e}_s$ be the unit vector tangent to the trajectory, and $\mathbf{n}$ the unit vector, normal to the boundary at the point of incidence; the direction of the launching angle is then defined by the unit vector $\mathbf{e}'_s$ given by

$$\mathbf{e}'_s = \mathbf{e}_s + \left(2\cos\theta'\right)\mathbf{n} \quad \text{and} \quad \cos\theta' = \mathbf{n}\cdot(-\mathbf{e}_s) \tag{8.18}$$

Eigenrays

Eigenrays are ray trajectories that connect the source to a given receiver. The calculation of eigenrays is a problem of extreme importance in Underwater Acoustics because it allows to produce predictions of *arrival times* at each receiver; however, it is also a problem of extreme complexity because the boundaries can induce elaborate types of reflection, in particular, by returning rays towards the source. Let us consider the particular case when a hydrophone is located at the position (r, z_h) and the final ray coordinate is given by $(r, z(\underline{\theta}))$; if the boundaries are "smooth" enough we can calculate the function

$$f(\underline{\theta}) = z(\underline{\theta}) - z_h$$

Then, by moving along $\underline{\theta}$ it is sufficient to determine each interval $\left[\underline{\theta}_i, \underline{\theta}_{i+1}\right]$ in which $f(\underline{\theta})$ changes sign and find the angle for which the function is zero (or at least smaller in absolute value than a given threshold); the solution defines the launching angle of an eigenray and its trajectory can be calculated by integrating the Eikonal equation. In the three-dimensional case, the calculation of eigenrays is more difficult even if the boundaries are smooth, because the three-dimensional position of the hydrophone does not allow to define a function that changes sign; optimization methods, such as the *simplex* method, have been found to be extremely efficient to deal with this problem [45].

The Isovelocity Waveguide

In the case of a waveguide with a depth D and a constant velocity of propagation c, the ray trajectories correspond to straight lines, which exhibit specular reflections on the surface and the bottom. For the positions $(0, z_s)$ of the source and (r_h, z_h) of an hydrophone, the launching angles of the eigenrays correspond to [48]

$$\begin{aligned}
\tan \underline{\theta}^{+}_{2m-1} &= \frac{(2m-2)D + z_s + z_h}{r_h} \\
\tan \underline{\theta}^{+}_{2m} &= \frac{2mD + z_s + z_h}{r_h} \\
\tan \underline{\theta}^{-}_{2m-1} &= \frac{2mD - (z_s + z_h)}{r_h} \\
\tan \underline{\theta}^{-}_{2m} &= \frac{2mD + z_h - z_s}{r_h}
\end{aligned} \tag{8.19}$$

where $\underline{\theta}^{+}$ and $\underline{\theta}^{-}$ represent launching angles towards the surface and the bottom, respectively, and m represents the number of reflections. For any arbitrary source–receiver configuration, the ray "skip" distance (that is, the horizontal distance between two consecutive surface or bottom reflections) is given by the expression:

$$D_{ray} = \frac{2D}{\tan \theta} \tag{8.20}$$

The arrival times of the eigenrays are given by the expressions

$$\begin{aligned}
\tau^{+}_{2m-1} &= \frac{\sqrt{r_h^2 + [z_s + z_h + (2m-2)D]^2}}{c} \\
\tau^{+}_{2m} &= \frac{\sqrt{r_h^2 + [z_s - z_h + 2mD]^2}}{c} \\
\tau^{-}_{2m-1} &= \frac{\sqrt{r_h^2 + [2mD - (z_s + z_h)]^2}}{c} \\
\tau^{-}_{2m} &= \frac{\sqrt{r_h^2 + [2mD + z_h - z_s]^2}}{c}
\end{aligned} \tag{8.21}$$

which in the general case represents a distribution of arrivals clustered in "quadruplets"; this pattern of clustering is often visible in the processing of real data in shallow

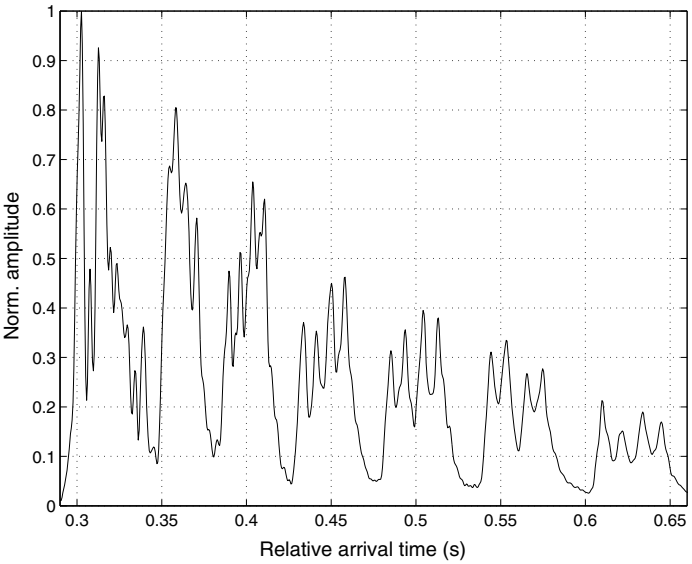


Fig. 8.1 Example of an arrival pattern, showing the clustering of arrivals in quadruplets (data from the INTIMATE96 experiment)

waters, even when the velocity of propagation is not constant (see Fig. 8.1). The symmetry of the solutions allows to infer the formation of "triplets" when $z_s = z_h \neq D/2$ and of "doublets" when $z_s = z_h = D/2$.

Numerical Example: The SOFAR Channel

The Eikonal admits analytical solutions in a limited number of cases; in the trivial case of a constant velocity, the trajectories correspond to straight lines, while in the case of a linear profile

$$c(z) = c_0 + gz$$

the trajectories correspond to circle arcs, with a curvature that corresponds to

$$R = \frac{1}{\xi g}$$

where $\xi = \cos \underline{\theta}(0)/c(0)$; an arbitrary profile, after numerical smoothing, can be approximated by segments with constant velocity or with a gradient; the ray trajectory can be calculated at each segment by resorting to analytical expressions [49]. Let us illustrate here the calculation of ray trajectories integrating the Eikonal given by Eq. (8.15) in the case of a Munk profile, for which we can write explicit partial derivatives of $c(z)$. This is done for a particular set of launching angles, using Python's

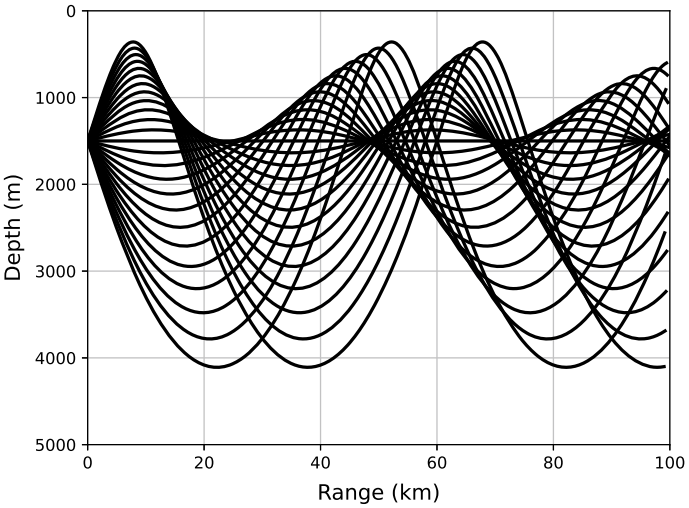


Fig. 8.2 Ray trajectories for the SOFAR channel

integrator of differential equations, which is numerically efficient but incompatible with boundary detection. This limitation can be overcome writing an integrator from scratch, which would introduce numerical approximations for the derivatives of an arbitrary sound speed profile, and will incorporate a boundary detector. This solution, however, would make the code exceptionally long and obscure, and would lead us apart from the fundamental goal, so we will sacrifice a full solution to the calculation of trajectories in favour of simplicity. The results of integration can be seen in Fig. 8.2, which is generated by the Python Code Snippet 8.1; as expected for the Munk profile, the propagation rays are "confined" to the SOFAR channel, a feature of propagation that supports the transmission of signals in deep waters at long distances. The results are also revealing the two fundamental characteristics of ray tracing, namely:

- The formation of *caustics*, which manifest in two different ways: the first, as the envelope of superimposing trajectories (well visible from 0 km up to 40 km), and the second as zones of ray convergence near 45, 70, and 90 km.
- The existence of *shadows*, that is, zones without rays, which in this case correspond to the intervals between depths from 0 to 300 m and from 4 to 5 km.

Ray tracing is numerically attractive because the algorithms that allow to deal with the Eikonal, in the general case, including boundary reflections, are numerically efficient (and in fact implemented a long time ago [50]). However, by itself, propagation rays would be of limited interest without predictions of acoustic pressure, which requires the solution of the transport equation. Before we proceed to discuss this issue it is important to describe an alternative method of calculation of ray trajectories.

Code Snippet 8.1 Integrating the Eikonal equation for the Munk profile.

```
from numpy import *
from scipy import integrate
from matplotlib.pyplot import *

ds = 100 # Ray step

Dmax = 5000
Rmaxkm = 100
Rmax = Rmaxkm*1000

# Source position:
rs = 0
zs = 1500

s = arange(rs,Rmax+ds,ds) # ray arclength

# Function for the Munk SSP:

z0 = 1500.0
c0 = 1480.0

def munk(z,z0,c0):
   B = 1.3e3
   e = 7.37e-3
   eta = 2*( z - z0 )/B
   c   = c0*( 1 + e*( eta + exp( -eta ) - 1 ) )
   dcdz = ( 2*c0*e/B  )*( 1 - exp( -eta ) )
   return c,dcdz

# Derivatives:

def solver(Y,s):
    r  = Y[0]
    z  = Y[1]
    sr = Y[2]
    sz = Y[3]
    c,dcdz = munk(z,z0,c0)
    cc = c*c
    dcdr = 0.0
    drds = c*sr
    dzds = c*sz
    dsrds = -dcdr/cc
    dszds = -dcdz/cc
    return [drds,dzds,dsrds,dszds]

# Define launching angles:

nrays = 25
thetas = linspace(-12,12,nrays)

for i in range(nrays):
    sr0 = cos( -thetas[i]*pi/180 )/c0
    sz0 = sin( -thetas[i]*pi/180 )/c0
    y0 = [rs, zs, sr0, sz0]
    # Integrate the ODE:
```

```
    y = integrate.odeint(solver, y0, s)

    r = squeeze( y[:,0] ); rkm = r/1000.0
    z = squeeze( y[:,1] )

    figure(1)
    plot(rkm,z,'k',linewidth=2)

xlabel('Range (km)',fontsize=18)
ylabel('Depth (m)',fontsize=18)
xlim(0,Rmaxkm)
ylim(Dmax,0)
grid(True)

show()
```

8.2.2 The Hamiltonian

The calculation of ray trajectories can be accomplished also relying on the concept of the *Hamiltonian*, which can be interpreted as a total energy of the system; the Hamiltonian depends on the so-called *conjugated* variables, akin to position and velocity, and the partial derivatives of the Hamiltonian along the conjugated variables define the evolution of the system; the calculation of trajectories can be optimized if the Hamiltonian is conveniently chosen. For instance, the Hamiltonian equivalent of Eq. (8.15) can be written as [3]

$$\frac{dp}{dr} = -\frac{\partial H}{\partial z}, \frac{dz}{dr} = \frac{\partial H}{\partial p} \tag{8.22}$$

with the particular choice

$$H = \frac{1}{2}\, p^2 + U \tag{8.23}$$

in which r and z represent the position variables, $p = \tan \underline{\theta}$ represents the "velocity", and the "potential energy" is defined as

$$U = \frac{1}{2}\left[1 - \frac{c_0^2}{c^2(z)}\right] \tag{8.24}$$

The partial derivatives can be written explicitly as

$$\frac{\partial H}{\partial p} = p \quad \text{and} \quad \frac{\partial H}{\partial z} = \frac{c_0^2}{c^3(z)}\,\frac{dc}{dz}$$

which has clear advantages from a numerical point of view. The calculation of trajectories resorting to the formalism of the Hamiltonian is illustrated in the Python Code Snippet 8.2, which in fact produces again Fig. 8.2. The Hamiltonian formalism is often used to characterize the chaos of ray trajectories when a random perturbation of sound speed is considered [51].

Code Snippet 8.2 Calculation of ray trajectories with the Hamiltonian formalism.

```
from numpy import *
from scipy import integrate
from matplotlib.pyplot import *

dr = 100 # Horizontal range step

Dmax = 5000
Rmaxkm = 100
Rmax = Rmaxkm*1000

# Source position:
rs = 0
zs = 1500

r = arange(rs,Rmax+dr,dr); rkm = r/1000.0

def solver(Y,r):
    p = Y[0]
    z = Y[1]
    B = 1.3e3
    epsilon = 7.37e-3
    z1 = 1500.0
    c1 = 1480.0
    eta = 2*( z - z1 )/B
    c   = c1*( 1 + epsilon*( eta + exp( -eta ) - 1 ) )
    dcdz = ( 2*c1*epsilon/B  )*( 1 - exp( -eta ) )
    ccc = c**3
    dHdz = c1*c1/ccc*dcdz
    dHdp = p
    return [-dHdz,dHdp]

# Define launching angles:

nrays = 25
thetas = linspace(-12,12,nrays)

for i in range(nrays):
    p0 = tan( -thetas[i]*pi/180 )
    z0 = zs
    y0 = [p0,z0]
    # Integrate the ODE:
    y = integrate.odeint(solver, y0, r)
    z = squeeze( y[:,1] )

    figure(1)
    plot(rkm,z,'k',linewidth=2)
```

```
xlabel('Range (km)',fontsize=18)
ylabel('Depth (m)' ,fontsize=18)
xlim(0,Rmaxkm)
ylim(Dmax,0)
grid(True)

show()
```

8.2.3 Geodesics

The discussion of propagation rays within the mathematical context of Tensor Calculus is rarely considered in the literature, but it is worthy of some attention given its elegance and generality. The starting point for the calculation of ray trajectories is given by the *geodesic equation*, which can be written as [52]

$$\ddot{y}_m = -\Gamma_{ij}^m \, \dot{y}_i \, \dot{y}_j \tag{8.25}$$

with $m = 1, 2, 3$; y_m represent the ray coordinates independently of the coordinate system; and Γ_{ij}^m represent the *Christoffel symbols* of the second kind, defined as

$$\Gamma_{ij}^m = \frac{1}{2} g^{mk} \left(\frac{\partial g_{ki}}{\partial x_j} + \frac{\partial g_{jk}}{\partial x_i} - \frac{\partial g_{ij}}{\partial x_k} \right) \tag{8.26}$$

in which g_{ij} defines the *Fermat's metric* of the system of coordinates, whose inverse defines the elements of the matrix g^{ij}; the definition given by Eq. (8.26) allows to see that $\Gamma_{ij}^m = \Gamma_{ji}^m$. In the discussion that follows for Eqs. (8.25) and (8.26) it will adopt the so-called *Einstein's rule*, in which the repetition of indexes represents a sum by the repeating indexes, that is,

$$c_j = a_i b_{ij} \quad \text{corresponds to} \quad c_j = \sum_{i=1}^{m} a_i b_{ij}$$

The analogy with a problem of kinematics should be evident: as in Newton's second law Eq. (8.25) defines an "acceleration" at any moment of time, and the calculation of a given trajectory requires also the definition of an "initial position" $y_m(0)$ (in this case the source position), and of an "initial velocity" $\dot{y}_m(0)$; this one is none other than the vector of propagation speed at the source position. For the particular case of Cartesian coordinates, one would have that

$$c = c(x, y, z)$$

$$(y_1(0), y_2(0), y_3(0)) = (x_s, y_s, z_s)$$

$$(\dot{y}_1(0), \dot{y}_2(0), \dot{y}_3(0)) = \mathbf{c}(x_s, y_s, z_s)$$

$$g_{ij} = \delta_{ij}/c^2, g^{ij} = c^2\delta_{ij}$$

where δ_{ij} represents Kronecker's delta symbol, and

$$\mathbf{\Gamma}^1 = \frac{1}{c}\begin{pmatrix} -\partial c/\partial x & -\partial c/\partial y & -\partial c/\partial z \\ -\partial c/\partial y & \partial c/\partial x & 0 \\ -\partial c/\partial z & 0 & \partial c/\partial x \end{pmatrix}$$

$$\mathbf{\Gamma}^2 = \frac{1}{c}\begin{pmatrix} \partial c/\partial y & -\partial c/\partial x & 0 \\ -\partial c/\partial x & -\partial c/\partial y & -\partial c/\partial z \\ 0 & -\partial c/\partial z & \partial c/\partial y \end{pmatrix}$$

$$\mathbf{\Gamma}^3 = \frac{1}{c}\begin{pmatrix} \partial c/\partial z & 0 & -\partial c/\partial x \\ 0 & \partial c/\partial z & -\partial c/\partial y \\ -\partial c/\partial x & -\partial c/\partial y & -\partial c/\partial z \end{pmatrix}$$

For instance, for the case when $c = c(z)$ the geodesic equation corresponds to

$$\frac{d^2x}{d\tau^2} = \frac{2}{c}\left(\frac{dc}{dz}\right)\left(\frac{dx}{d\tau}\right)\left(\frac{dz}{d\tau}\right)$$

and

$$\frac{d^2z}{d\tau^2} = \frac{1}{c}\left(\frac{dc}{dz}\right)\left[\left(\frac{dz}{d\tau}\right)^2 - \left(\frac{dx}{d\tau}\right)^2\right]$$

which, at a first sight, seem to be completely unrelated to the Eikonal equation:

$$\frac{d^2\mathbf{r}}{ds^2} = \frac{dc}{ds}\boldsymbol{\sigma} - \frac{1}{c}\nabla c$$

where

$$\mathbf{r} = \mathbf{e}_x x + \mathbf{e}_z z \quad \text{and} \quad \boldsymbol{\sigma} = \frac{1}{c}\left(\frac{dx}{ds}\mathbf{e}_x + \frac{dz}{ds}\mathbf{e}_z\right)$$

However, it is possible to demonstrate that the Eikonal equation becomes the geodesic equation by taking into account that

$$\frac{dc}{ds} = \frac{1}{c}\left(\frac{dc}{dz}\right)\left(\frac{dz}{d\tau}\right), \boldsymbol{\sigma} = \frac{1}{c^2}\left(\frac{dx}{d\tau}\mathbf{e}_x + \frac{dz}{d\tau}\mathbf{e}_z\right)$$

and

$$\frac{d^2x}{ds^2} = \frac{d}{ds}\left(\frac{dx}{ds}\right) = -\frac{1}{c^2}\left(\frac{dc}{ds}\right)\left(\frac{dx}{d\tau}\right) + \frac{1}{c^2}\frac{d^2x}{d\tau^2}$$

together with

$$\frac{d^2z}{ds^2} = \frac{d}{ds}\left(\frac{dz}{ds}\right) = -\frac{1}{c^2}\left(\frac{dc}{ds}\right)\left(\frac{dz}{d\tau}\right) + \frac{1}{c^2}\frac{d^2z}{d\tau^2}$$

The calculation of ray trajectories for the Munk profile resorting to the geodesic formalism is illustrated numerically in the Python Code Snippet 8.3, which in fact produces again Fig. 8.2.

Code Snippet 8.3 Calculation of ray trajectories as geodesics.

```
from numpy import *
from scipy import integrate
from matplotlib.pyplot import *

Dmax = 5000
Rmaxkm = 100
Rmax = Rmaxkm*1000

# Source position:
rs = 0
zs = 1500

dtau = 0.1
t = arange(0,67.0+dtau,dtau)

z0 = 1500.0
c0 = 1480.0

# Munk SSP:
def munk(z,z0,c0):
    B = 1.3e3
    e = 7.37e-3
    eta = 2*( z - z0 )/B
    c   = c0*( 1 + e*( eta + exp( -eta ) - 1 ) )
    dcdz = ( 2*c0*e/B  )*( 1 - exp( -eta ) )
    return c,dcdz

# Derivatives:
def solver(Y,s):
     r  = Y[0]
     z  = Y[1]
     X1 = Y[2]
     X2 = Y[3]
     c,dcdz = munk(z,z0,c0)
     X1t = ( 2.0/c )*dcdz*X1*X2
     X2t = ( 1.0/c )*dcdz*( X2*X2 - X1*X1 )
     return [X1,X2,X1t,X2t]
```

```
# Launching angles:
nrays = 25
thetas = linspace(-12,12,nrays)

for i in range(nrays):
    vr = c0*cos( -thetas[i]*pi/180 )
    vz = c0*sin( -thetas[i]*pi/180 )
    y0 = [rs, zs, vr, vz]
    # ODE integration:
    y = integrate.odeint(solver, y0, t)

    r = squeeze( y[:,0] ); rkm = r/1000.0
    z = squeeze( y[:,1] )

    figure(1)
    plot(rkm,z,'k',linewidth=2)

xlabel('Range (km)',fontsize=18)
ylabel('Depth (m)',fontsize=18)
xlim(0,Rmaxkm)
ylim(Dmax,0)
grid(True)

show()
```

8.2.4 Ray Amplitudes

Ray trajectories can be used for the calculation of ray *tubes* and of the associated *Jacobian*, which is nothing more than the tube cross section. Representing by $\mathbf{e}_s$ the tangent vector to the tube central ray, we can rewrite the first of the terms in Eq. (8.3) as

$$\nabla A \cdot \nabla \tau = \frac{dA}{ds}\mathbf{e}_s \cdot \frac{d\tau}{ds}\mathbf{e}_s = \frac{dA}{ds}\mathbf{e}_s \cdot \frac{1}{c}\mathbf{e}_s = \frac{1}{c}\frac{dA}{ds}$$

For the second term, we can resort to the properties of the Jacobian to write that [3]

$$\nabla^2 \tau = \nabla \cdot \nabla \tau = \frac{1}{J}\frac{d}{ds}\left(\frac{J}{c}\right)$$

which transforms the transport equation into

$$\frac{2}{c}\frac{dA}{ds} + \frac{A}{J}\frac{d}{ds}\left(\frac{J}{c}\right) = 0 \tag{8.27}$$

which can be easily integrated to provide the result

$$A = \frac{A_0}{\sqrt{J/c(s)}}$$

in which the constant A_0 depends on the acoustic source. The complete solution for pressure along the ray can then be written as

$$P(s,\omega) = A_0\sqrt{\frac{c(s)}{J}}\, e^{-i\omega\tau}$$

In the proximity of a point source, we can write that

$$c(s) \approx c(0) \quad \text{and} \quad J \approx s^2 \cos \underline{\theta}(0)$$

Comparing the last two expressions

$$A_0\sqrt{\frac{c(0)}{s^2\cos\theta(0)}}\, e^{-i\omega s/c(0)} = \frac{1}{4\pi s}\, e^{-i\omega s/c(0)};$$

we can conclude that

$$A_0 = \frac{1}{4\pi}\sqrt{\frac{\cos \underline{\theta}(0)}{c(0)}}$$

allowing to write conclusively the ray pressure as

$$P(s,\omega) = \frac{1}{4\pi}\sqrt{\frac{c(s)}{c(0)}\frac{\cos \underline{\theta}(0)}{J}}\, e^{-i\omega\tau} \tag{8.28}$$

The coefficient $1/(4\pi)$ is irrelevant for the calculation of transmission loss because the reference pressure for $r = 1$ m can be represented as

$$P_{ref} = \frac{1}{4\pi}$$

The calculation of amplitudes should include the effects of attenuation along distance (see Sect. 4.5); in the case of rays being reflected on a boundary, the amplitude should be multiplied by the reflection coefficient (see Sect. 5.7).

The calculation of acoustic pressure based on Eq. (8.28) has two limitations:

- It is valid in ray coordinates; in order to calculate the pressure at a given point of the chosen coordinate system, one needs to calculate the eigenrays for each position. Alternatively, ray coordinates can be mapped to the system coordinates, and this mapping can be used to determine the pressure field resulting from the interference of the rays that coincide at the given point.
- The Jacobian can be approximated as

$$J \approx r\left(\frac{dz}{d\underline{\theta}}\right)$$

in which r represents the horizontal distance travelled by the tube central ray, dz is the vertical separation between rays, and $d\underline{\theta}$ represents the difference between the grazing angles of the rays. At the turning points dz together with $d\underline{\theta}$ tend to zero, but their ratio remains finite; at the acoustics

$$\frac{dz}{d\underline{\theta}} \approx 0$$

and the acoustic pressure suddenly increases, manifesting itself in a sudden decrease of the transmission loss.

Such drawbacks can be overcome considering that each ray represents the central axis of a Gaussian beam, whose *influence* at the point of interest represents the contribution of the ray to the total pressure field; Gaussian beams allow also to take caustics into account without the drastic effect of amplitude increasing suddenly [41, 52, 53].

8.2.5 *Ray Interference*

Let us discuss now a case of ray interference that admits an exact solution when the velocity of propagation is constant. Let us consider a source at the position $(0, z_s)$ and a receiver at the position (r_h, z_h), at which arrive a direct ray and a surface reflected ray (see Fig. 8.3); an important boundary condition for the surface is that pressure should be equal to zero. Thus, the pressure field can be written as [3]

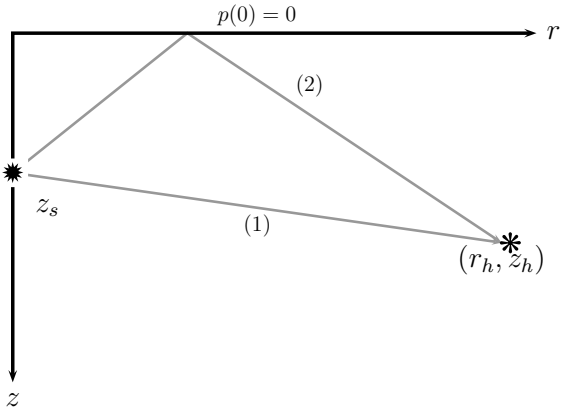


Fig. 8.3 Direct ray and surface reflected ray

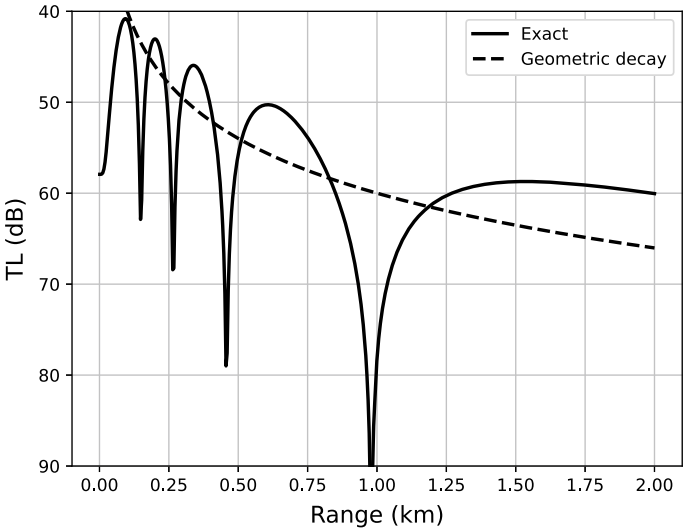


Fig. 8.4 Ray interference (Lloyd mirror)

$$p(r_h, z_h) = \frac{e^{ikR_1}}{R_1} - \frac{e^{ikR_2}}{R_2} \tag{8.29}$$

where

$$R_1 = \sqrt{r_h^2 + (z_h - z_s)^2} \quad \text{and} \quad R_2 = \sqrt{r_h^2 + (z_h + z_s)^2}$$

The simplicity of this expression hides a complex interference pattern (as long as the surface is a perfect reflector) known as *Lloyd's mirror*, which converges rapidly to a geometric decay (see Fig. 8.4, generated by the Python Code Snippet 8.4). Such interference pattern can be used for the localization of a moving source [54].

Code Snippet 8.4 Ray interference (Lloyd mirror).

```
from numpy import *
from matplotlib.pyplot import *

cw   = 1500.0
freq = 150.0
Tx   = 25.0
Rx   = 200.0
rmin =    0.0
rmax = 2000.0
nr = 501
r = linspace(rmin,rmax,nr)
r[0] = 1.0
rkm = r/1000
```

```
l = cw/freq
k = 2*pi/l
R1 = sqrt( r*r + ( Tx - Rx )*( Tx - Rx ) )
R2 = sqrt( r*r + ( Tx + Rx )*( Tx + Rx ) )

p = exp( 1j*k*R1 )/R1 - exp( 1j*k*R2 )/R2

TL  = -20*log10( abs( p ) ) # Ray TL
TLg =  20*log10( r ) # TL geometric decay

figure(1)
plot(rkm,TL ,'k'  ,linewidth=2,
label='Exact')
plot(rkm,TLg,'k--',linewidth=2,
label='Geometric decay')
xlabel('Range (km)',fontsize=18)
ylabel('TL (dB)',fontsize=18)
ylim(90,40)
legend(loc=1)
grid(True)

show()
```

8.2.6 *Source Directivity*

In the discussion presented in the previous sections, it is implicit the emission of rays by a point source, with the pressure field near the source corresponding to the field of a spherical wave; under such conditions the initial ray amplitudes are independent of the direction of propagation. Real sources, however, can exhibit specific patterns of emission, which give predominance to some directions in detriment of others. Not all ray tracing models allow to include the directivity of a real source, and the fact is that such directivity seldom is noticed in the literature of Underwater Acoustics. If relevant the pattern of source emission can be easily incorporated into ray tracing models, just requiring for initial ray amplitudes to be weighted according to the source directivity.

8.3 Normal Modes

Let us consider another solution to Eq. (4.2), which is now represented as [29]

$$P(r, z) = Z(z)\ R(r) \tag{8.30}$$

After substitution we can obtain the following differential equation for the vertical function:

$$\frac{d^2Z}{dz^2} + \left(\frac{\omega^2}{c^2(z)} - \underline{k}^2\right) Z = 0 \tag{8.31}$$

in which $\underline{k}$ represents a separation constant between $Z(z)$ and $R(r)$. The above equation, together with a proper choice of boundary conditions, represents a classic *Sturm–Liouville* problem [55] with well-known properties:

- The differential equation has an infinite number of solutions, denominated propagation *modes* $Z_m(z)$, which correspond to the eigenfunctions of the Sturm–Liouville problem.
- For each eigenfunction, there is a corresponding eigenvalue $\underline{k}_m^2$.
- Each eigenfunction $Z_m(z)$ has m zeros in the interval $z \in [0, D]$.
- If $D < \infty$ the Sturm–Liouville problem is said to be *regular*, and the $\underline{k}_m^2$ can be organized in such a way that
$$\underline{k}_1^2 > \underline{k}_2^2 > \cdots$$
- It can be shown that
$$\left|\underline{k}_m\right| < \frac{\omega}{c_{\min}}$$
$\forall m$, where $c_{\min} = \min c(z)$.

The modes can be normalized in order to form a space of orthogonal eigenfunctions:

$$\int_0^D Z_m(z)\ Z_n(z)\ dz = \delta_{mn} \tag{8.32}$$

Therefore, the pressure field can be represented on the basis of modes as

$$P(r, z) = \sum_{m=1}^{\infty} Z_m(z)\ R_m(r) \tag{8.33}$$

which allows to obtain the following equation for an impulsive source:

$$\sum_{m=1}^{\infty} \left[\frac{1}{r}\frac{d}{dr}\left(r\frac{dR_m}{dr}\right) Z_m + R_m\left(\frac{d^2Z_m}{dz^2} + \frac{\omega^2}{c^2}Z_m\right)\right] = \frac{\delta(r)\delta(z - z_s)}{2\pi r} \tag{8.34}$$

Since[3]

$$\frac{d^2Z_m}{dz^2} + \frac{\omega^2}{c^2(z)}Z_n = \underline{k}_m^2 Z_m$$

[3] If density is to be taken into account the modal differential equation should be written as

$$\rho(z)\frac{d}{dz}\left(\frac{1}{\rho(z)}\frac{dZ_m}{dz}\right) + \left(\frac{\omega^2}{c^2(z)} - \underline{k}_m^2\right) Z_m = 0$$

.

we can rewrite Eq. (8.34) as

$$\sum_{m=1}^{\infty}\left[\frac{1}{r}\frac{d}{dr}\left(r\frac{dR_m(r)}{dr}\right)+\underline{k}_m^2 R_m(r)\right]Z_m(z)=-\frac{\delta(r)\delta(z-z_s)}{2\pi r} \tag{8.35}$$

Applying now the operator

$$\int_0^D (\,\cdot\,)\,Z_l(z)\,dz$$

to both sides of the equation we can obtain that

$$\frac{1}{r}\frac{d}{dr}\left(r\frac{dR_l(r)}{dr}\right)+\underline{k}_l^2 R_l=-\frac{\delta(r)Z_l(z_s)}{2\pi r} \tag{8.36}$$

whose standard solution, according to the *radiation condition*,

$$R(r)\to 0 \quad \text{when} \quad r\to\infty$$

is given by

$$R_l(r)=\frac{i}{4}Z_l(z_s)H_0^{(1)}\left(\underline{k}_l r\right)\ , \tag{8.37}$$

in which $H_0^{(1)}(\underline{k}_l r)$ is the Hankel function of the first kind; for large distances its asymptotic behaviour corresponds to [55]

$$H_0(r)\approx\sqrt{\frac{2}{\pi r}}\,e^{i(r-\pi/4)}$$

Under these conditions we can write the final expression for the pressure field on the modal basis as [29]

$$P(r,z)=\frac{i}{\sqrt{8\pi}}\,e^{-i\pi/4}\sum_{m=1}^{\infty}Z_m(z_s)\;Z_m(z)\frac{e^{i\underline{k}_m r}}{\sqrt{\underline{k}_m r}} \tag{8.38}$$

As in the case of ray tracing one needs to include the effects of attenuation over distance. On another side, as shown by the modal expansion (and in sharp contrast with ray theory), there is a chance to "prevent" the excitation of a given mode (or group of modes) through a choice of z_s, such that $Z_m(z_s)=0$. This is an issue of practical interest for waveguides that exhibit a *thermocline* (that is, a sudden decrease of the propagation speed as depth increases); in an environment with such characteristics, the source can only excite modes of lower order if located below the thermocline.

For the calculation of transmission loss, we can resort to the expression

$$\mathrm{TL} = -20 \log \left[\frac{P(r,z)}{P_0} \right]$$

where

$$P_0 = \frac{1}{4\pi}$$

From the point of view of Physics, Eq. (8.38) corresponds to an approximation of the pressure field as a superposition of individual fields:

$$P \approx \sum_m P_m$$

in which each P_m corresponds to a stationary wave, multiplied by a propagating cylindrical wave:

$$P_m = \left(e^{i\hat{k}_m z} + e^{-i\hat{k}_m z} \right) \frac{e^{i\underline{k}_m r}}{\sqrt{\underline{k}_m r}}$$

In this expression, $\hat{k}$ represents the "vertical" wavenumber and $\underline{k}$ is the "horizontal" wavenumber:

$$\hat{k}^2 + \underline{k}^2 = \frac{\omega^2}{c^2}$$

An analogy with ray theory can be introduced defining the *skip distance* D_m as given by [56]

$$D_m = -2\pi \left(\frac{\partial m}{\partial \underline{k}_m} \right)_\omega \tag{8.39}$$

where D_m represents the horizontal distance travelled by a ray, "equivalent" to the mode, between consecutive turning points.

In the general case, we can associate a phase velocity with each mode, which is given by

$$v_p^m = \frac{\omega}{\underline{k}_m}$$

and also a group velocity

$$v_g^m = \frac{d\omega}{d\underline{k}_m}$$

which defines a *modal time* of propagation as

$$\tau_m = \frac{r}{v_g^m}$$

As shown by the previous expression the modal time depends on the horizontal distance only, in sharp contrast with the ray propagation time, which is defined along a ray trajectory. The modal functions can be used also for the calculation of group velocity [56]; for the case of a constant density and a finite depth, it can be written as

$$v_g^m = \frac{1}{v_p^m} \frac{p_m}{q_m}$$

where

$$p_m = \int_0^D Z_m^2(z)\, dz$$

and

$$q_m = \int_0^D \frac{Z_m^2(z)}{c^2(z)}\, dz$$

As will be shown in the following section the modal expansion can be calculated accurately with a limited number of modes.

8.3.1 The Isovelocity Waveguide

Let us discuss the calculation of modes in an isovelocity waveguide with constant depth D and constant speed of propagation c; we will consider the boundary condition

- $Z(0) = 0$, which implies that $P(0, r) = 0$; under such condition the surface is called a *pressure release boundary*;
- $dZ/dz = 0$ for $z = D$; the boundary under this condition is known to be *rigid*.

The normalized solution of Eq. (8.31), associated to the given set of boundary conditions, corresponds to [29]

$$Z_m(z) = \sqrt{\frac{2}{D}} \sin\left(\hat{k}_m z\right)$$

in which

$$\hat{k}_m = \left(m - \frac{1}{2}\right) \frac{\pi}{D}$$

with $m = 1, 2, 3, \ldots$, and

$$\underline{k}_m = \sqrt{\frac{\omega^2}{c^2} - \hat{k}_m^2}$$

The calculation of transmission loss can take place using the expression

$$\mathrm{TL} = \left| \frac{\sqrt{8\pi}}{D} \sum_m \sin(\hat{k}_m z_s) \sin(\hat{k}_m z) \frac{e^{i\underline{k}_m r}}{\sqrt{\underline{k}_m r}} \right|$$

What is interesting in this case is that the vertical wavenumbers $\hat{k}_m$ are frequency *independent*, thus *all* sources excite the same set of stationary waves in the water-column; however, the same does not apply to the propagating waves: the horizontal wavenumbers $\underline{k}_m$ are all positive up to the index

$$m_c < \left(\frac{\omega D}{\pi c} + \frac{1}{2} \right)$$

while the remaining ones are imaginary; therefore, the value of frequency divides the modes into two groups, namely:

- the *trapped modes*, with real $\underline{k}_m$ and
- the *evanescent modes*, with imaginary $\underline{k}_m$; for these ones the propagating wave exhibits an exponential decay in amplitude as the horizontal distance increases.

To these two groups we need to add a third one: the group of *leaky modes*, for which $\underline{k}_m$ can be complex. Depending on the specific value of frequency m_c can take the values 1, 2, 3, etc., defining a *cut-off frequency* for each mode; in some cases, the modal expansion requires only the superposition of trapped modes.

We can take advantage of the explicit expressions for the wavenumbers to show that the phase velocity and the group velocity correspond to

$$v_p^m = \frac{\omega}{\underline{k}_m} = \frac{\omega}{\sqrt{\dfrac{\omega^2}{c^2} - \left[\left(m - \dfrac{1}{2} \right) \dfrac{\pi}{D} \right]^2}}$$

and

$$v_g^m = c^2 \frac{\underline{k}_m}{\omega} \Rightarrow v_g^m v_p^m = c^2$$

On another side after some algebra we can find also that the mode skip distance corresponds to

$$D_m = \frac{2D}{\tan\theta_m} \quad \text{where} \quad \tan\theta_m = \frac{\hat{k}_m}{\underline{k}_m}$$

with D_m coinciding with the ray skip distance defined by Eq. (8.20).

The behaviour of phase and group velocities along frequency is shown in Fig. 8.5, and reveals a surprising trend of opposites: in both cases, the velocities move away from the asymptote ω/c and the separation between curves decreases as frequency increases; yet, the phase velocities *increase* progressively as the mode order increases,

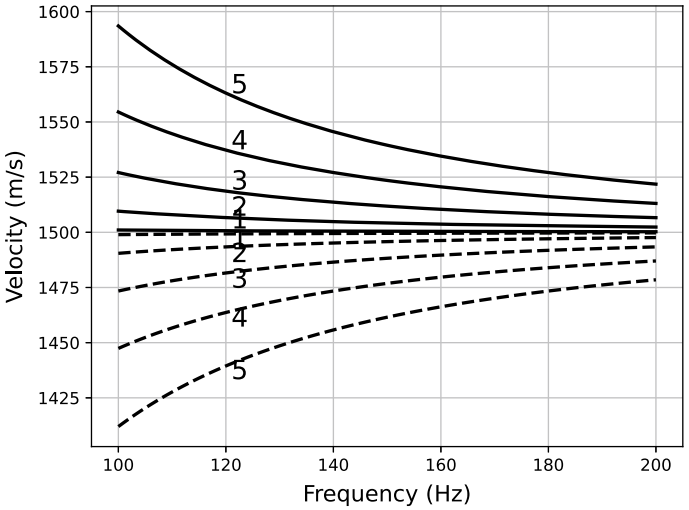


Fig. 8.5 Isovelocity waveguide: phase velocity (continuous line) and group velocity (dashed line) as a function of frequency; the mode order is indicated next to each curve

while group velocities *decrease*. As discussed in Sect. 5.4 this differentiation of group velocities implies that the modes arrive progressively at separate instants at the receiver, with the arrival order being defined by the mode order. This is known as *temporal* dispersion.

The issue of the temporal order of arrival is not the only type of dispersion that affects the propagation of an acoustic signal, because the travel modal time is also a function of frequency; in the general case, each modal group velocity increases with frequency, meaning that for a given instant higher frequencies propagate faster than lower frequencies. This modal dispersion manifests itself in spectrograms of explosive signals as curves in time and frequency, in which each curve corresponds to a mode of propagation. The compensation of dispersion effects can be used to determine the distance between the source and the reception system; to this end one needs to model the impulse response resorting to the modal solution [57]. For the isovelocity case such modelling can take place in the following manner: the functional dependence between the wavenumbers and frequency can be written explicitly as

$$\underline{k}_m(\omega) = \frac{\sqrt{\omega^2 - \omega_{0m}^2}}{c}$$

where

$$\omega_{0m} = (m - 1/2)\,\pi c/D$$

represents the cut-off frequency. For a given distance, and without loss of generality, it is possible to write the modal expansion in the form

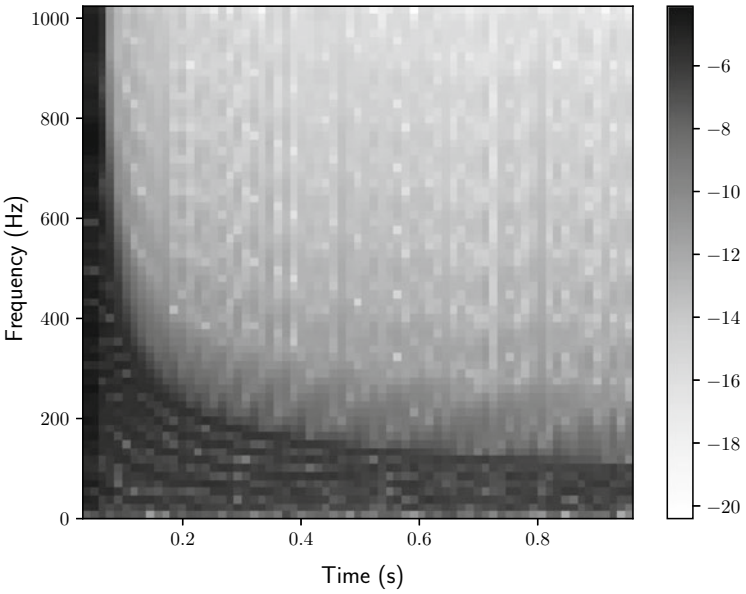


Fig. 8.6 Spectrogram of an impulse response showing modal dispersion

$$H = \sum_m |H_m| \, e^{-i\underline{k}_m r}$$

where the negative sign is introduced in order to take into account that from the reception point of view the modes are delayed in time; the impulse response can then be obtained through Fourier synthesis. In order to control the initial position of the impulse response we can multiply H by a factor $e^{i\omega t_0}$, where $t_0 = r/c$. In the general case, the wavenumbers should be calculated for the desired frequency band. The procedure above described is illustrated in Fig. 8.6, which shows a typical pattern of dispersion for modes 1–5; the figure is generated by the Code Snippet 8.5 for the parameters $D = 100\,\text{m}$, $c = 1500\,\text{m/s}$ and $r = 20\,\text{km}$; for simplicity, it is considered that $|H_m| = 1$. The effects of dispersion on the temporal record of a signal can be analysed in detail based on special processing techniques, such as *Time Warping* [58].

Code Snippet 8.5 Modal dispersion (adapted from [57]).

```
from scipy import signal
from numpy import *
from matplotlib.pyplot import *

rcParams.update({''text.usetex'': True})
```

```
sampling = 2048
h = zeros( sampling )
nyquist = int( sampling/2 )
freqs = arange(0,sampling,1)
dt = 1.0/sampling
t = arange(0,1,dt)
D = 100.0
Rkm = 20.0
R = Rkm*1000.0
c  = 1500.0
cxc = c*c
# Time delay to avoid start at t = 0:
t0 = R/c
t0 = ( int( t0/dt ) - 50 )*dt
for m in range(1,6):
    H  = ones(sampling) + 1j*zeros(sampling)
    w  = 2*pi*freqs
    # Cut-off frequency:
    f0m = (m - 0.5)*c/(2*D)
    w0m = 2*pi*f0m
    # Wavenumber square:
    kxk = ( w**2 - w0m**2 )/cxc
    # Find evanescent modes:
    indexes = argwhere( kxk < 0 )
    L = indexes.size
    # Remove evanescent modes:
    kxk[ indexes ] = 0.0
    w[   indexes ] = 0.0
    k = sqrt( kxk )
    H[ indexes ] = 0.0
    H[ sampling-indexes-1 ] = 0.0
    # Impulse response:
    phase = R*k - w*t0
    H = H*exp( -1j*phase )
    h = h + real( fft.ifft( H ) )
fe, te, Shh = signal.spectrogram(h,
sampling, nfft=128, nperseg=128, noverlap=100)
figure(1)
plot(t,h)
xlabel('Time (s)',fontsize=18)
title(r'Channel impulse response $h(t)$',
fontsize=18)
grid(True)
figure(2)
imshow(log10(abs(Shh)), cmap='Greys',
extent=[te[0],te[-1],fe[0],fe[-1]],
aspect='auto',origin='lower')
xlabel('Time (s)',fontsize=18)
ylabel('Frequency (Hz)',fontsize=18)
colorbar()
show()
```

8.3.2 The Pekeris Waveguide

Let us consider another case of mode calculations, this time for the *Pekeris* waveguide, which corresponds to the case of a watercolumn with depth D, water sound speed c_1, and density ρ_1, over an infinite sediment with density ρ_2 and sound speed c_2 (see Fig. 8.7).

The modal function can be represented as [29]

$$Z(z) = \begin{cases} Z_1(z) = A \sin\left(\hat{k}_1 z\right) & \text{for } z < D \\ Z_2(z) = B \exp\left(-\hat{k}_2 z\right) & \text{for } z > D \end{cases}$$

where

$$\hat{k}_1 = \sqrt{\left(\frac{\omega}{c_1}\right)^2 - \underline{k}^2} \quad \text{and} \quad \hat{k}_2 = \sqrt{\underline{k}^2 - \left(\frac{\omega}{c_2}\right)^2}$$

The boundary conditions correspond to

- Surface pressure should be zero, thus $Z_1(0) = 0$.
- The modal function should be continuous at the boundary:

$$Z_1(D) = Z_2(D).$$

- The vertical component of particle velocity should be continuous too:

$$\left(\frac{1}{\rho_1}\frac{dZ_1}{dz}\right)_{z=D} \left(\frac{1}{\rho_2}\frac{dZ_2}{dz}\right)_{z=D}.$$

- For obvious reasons $Z_2(z) \to 0$ when $z \to \infty$, which transforms the Pekeris case into a singular Sturm–Liouville problem.

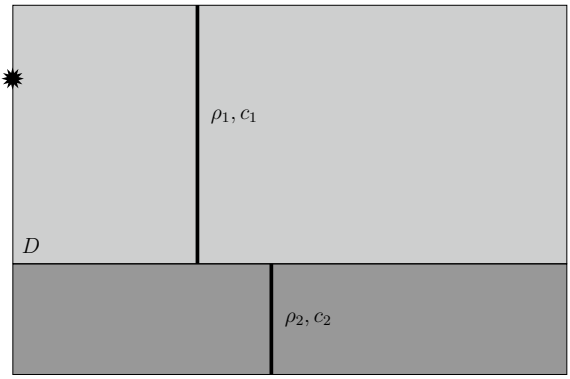


Fig. 8.7 Pekeris waveguide

The boundary conditions can only be satisfied by the horizontal wavenumbers that correspond to the zeros of the *characteristic equation*; in the specific case of the Pekeris problem, the equation can be written as

$$f(\underline{k}) = \frac{\underline{S}_1 \underline{R}_2}{\rho_2} + \frac{\underline{C}_1 \underline{R}_1}{\rho_1}$$

where

$$\underline{S}_1 = \sin\left(\underline{R}_1 D\right) \quad , \quad \underline{C}_1 = \cos\left(\underline{R}_1 D\right)$$

and

$$\underline{R}_1 = \sqrt{\left(\frac{\omega}{c_1}\right)^2 - \underline{k}^2} \quad , \quad \underline{R}_2 = \sqrt{\underline{k}^2 - \left(\frac{\omega}{c_2}\right)^2}$$

For the given conditions, the cut-off frequency corresponds to [57]:

$$\omega_{0m} = (m - 1/2)\,\frac{\pi c_1}{D}\sqrt{\frac{1}{1 - (c_1/c_2)^2}}$$

The dispersion curves for the Pekeris waveguide are shown in Fig. 8.8, which is generated by the Python Code Snippet 8.6; it is possible to see that the phase velocities behave like in the previous case, but the particular choice of frequencies allows to show how higher order modes are being excited progressively after each mode's cut-off frequency. Group velocities, on the other side, exhibit an unexpected

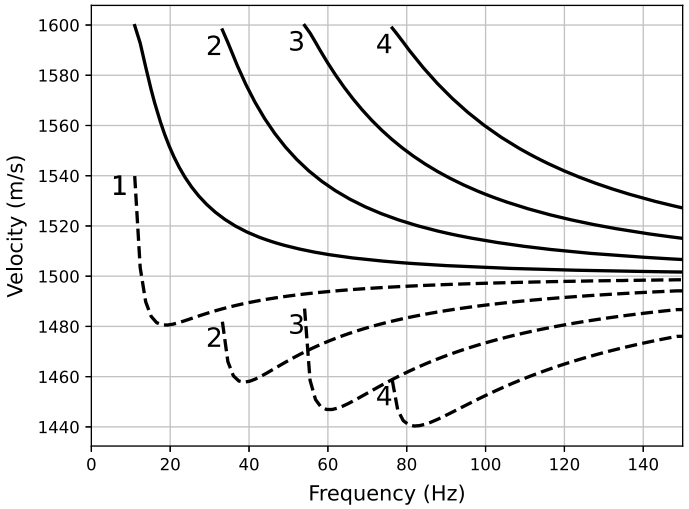


Fig. 8.8 Pekeris waveguide: phase velocity (continuous lines) and group velocity (dashed lines) as a function of frequency; the mode order is indicated next to each curve

behaviour by the fact that they decrease after the cut-off frequency, achieve a minimal in which $dv_g/d\omega = d^2\underline{k}/d\omega^2 = 0$ and then "recover" their asymptotic behaviour along frequencies; this is called an *anomalous* dispersion.

Code Snippet 8.6 Pekeris waveguide: dispersion curves.

```
from numpy import *
from matplotlib.pyplot import *

def kpekeris(c1 = None, rho1 = None,
             c2 = None, rho2 = None,
             D = None, freq = None):

    kp = []

    w = 2*pi*freq
    k1 = w/c1
    k2 = w/c2

    nk = 1001
    k = linspace(k1,k2,nk)
    kxk = k*k

    r1 = sqrt( k1*k1 - kxk )
    r2 = sqrt( kxk - k2*k2 )

    s1 = sin( r1*D )
    c1 = cos( r1*D )

    f = s1*r2/rho2 + c1*r1/rho1

    for i in range(nk-1):

        x1 = k[ i ]
        x2 = k[i+1]
        y1 = f[ i ]
        y2 = f[i+1]
        p = y1*y2
        x3  = (x1*y2-x2*y1)/(y2-y1)

        if p < 0:

           kp = append(kp,x3)

    return kp

D    =  100.0
rho1 =    1.0
rho2 =    1.5
c1   = 1500.0
c2   = 1600.0
nfreqs = 101
fmin = 11.0
fmax = 150.0
freqs = linspace(fmin,fmax,nfreqs)
```

```
w = 2*pi*freqs

k1 = zeros(nfreqs)
k2 = zeros(nfreqs) + nan
k3 = zeros(nfreqs) + nan
k4 = zeros(nfreqs) + nan

mc2 = nfreqs
mc3 = nfreqs
mc4 = nfreqs

for i in range(nfreqs):
    wi = 2*pi*freqs[i]
    kpek = kpekeris(c1,rho1,c2,rho2,D,freqs[i])
    k1[i] = kpek[0]
    nk = kpek.size
    if nk == 2:
       k2[i] = kpek[1]
       mc2 = min([i,mc2])
    if nk == 3:
       k2[i] = kpek[1]
       k3[i] = kpek[2]
       mc3 = min([i,mc3])
    if nk >= 4:
       k2[i] = kpek[1]
       k3[i] = kpek[2]
       k4[i] = kpek[3]
       mc4 = min([i,mc4])

dw = diff( w )
vp1 = w/k1
vg1 = dw/diff(k1); vg1 = append(vg1,vg1[-1])
vp2 = w/k2
vg2 = dw/diff(k2); vg2 = append(vg2,vg2[-1])
vp3 = w/k3
vg3 = dw/diff(k3); vg3 = append(vg3,vg3[-1])
vp4 = w/k4
vg4 = dw/diff(k4); vg4 = append(vg4,vg4[-1])

figure(1)
plot(freqs,vp1,'k'  ,linewidth=2)
plot(freqs,vg1,'k--',linewidth=2)
plot(freqs,vp2,'k'  ,linewidth=2)
plot(freqs,vg2,'k--',linewidth=2)
plot(freqs,vp3,'k'  ,linewidth=2)
plot(freqs,vg3,'k--',linewidth=2)
plot(freqs,vp4,'k'  ,linewidth=2)
plot(freqs,vg4,'k--',linewidth=2)
text(5,vg1[0]-7.5,'1',fontsize=16)
text(5,vg1[0]-7.5,'1',fontsize=16)
text(freqs[mc2-3],vp2[mc2]-10,'2',fontsize=16)
text(freqs[mc2-3],vg2[mc2]-10,'2',fontsize=16)
text(freqs[mc3-3],vp3[mc3]-10,'3',fontsize=16)
text(freqs[mc3-3],vg3[mc3]-10,'3',fontsize=16)
text(freqs[mc4-3],vp4[mc4]-10,'4',fontsize=16)
text(freqs[mc4-3],vg4[mc4]-10,'4',fontsize=16)
xlabel('Frequency (Hz)',fontsize=18)
ylabel('Velocity (m/s)',fontsize=18)
```

```
xlim(0,fmax)
grid(True)

show()
```

8.3.3 Airy Phase

Let us consider the temporal propagation of an arbitrary mode; by the inverse Fourier transform we can write that [59]

$$p_m(r,t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} P_m(r,\omega)\; e^{-i\left(\omega t - \underline{k}_m r\right)} d\omega \tag{8.40}$$

For a non-dispersive mode, one would have that

$$p_m(r,t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} P_m(r,\omega)\; e^{-i\omega(t-t_m)} d\omega = p_m(t-t_m) \tag{8.41}$$

where $t_m = r/v_p^m$ and $v_p^m = \omega/\underline{k}_m$, meaning that the received signal is just a delayed "replica" of the emitted signal. For a dispersive mode, we can resort to the *stationary phase approximation* [55], which considers only the contribution of the terms around a "dominant" frequency, allowing to write that

$$-\omega t + \underline{k}_m r \approx -\omega_0 t + \underline{k}_m(\omega_0) r + \frac{r}{2}\,\frac{d^2\underline{k}_m}{d\omega^2}\,(\omega-\omega_0)^2$$

Under these conditions, the mode can be approximated in the temporal domain through the expression

$$p_m(r,t) \approx \frac{|P_m(r,\omega_0)|}{\pi}\sqrt{\frac{2\pi}{r\left|d^2\underline{k}/d\omega^2\right|}}\;\cos\left[-\omega_0 t + \underline{k}_m(\omega_0) r \pm \pi/4\right] \tag{8.42}$$

in which the sign to be used should coincide with the sign of the second derivative. This expression is obviously invalid in the cases when

$$\frac{d^2\underline{k}_m}{d\omega^2} = 0$$

which defines the *Airy phase*, so denominated because the approximation $-\omega t + \underline{k}_m r$ requires terms of higher order for which it is possible to obtain expressions of $p_m(r, t)$ involving Airy functions. The Airy phase is obviously related with the minima of group velocity and anomalous dispersion, discussed in the previous section; in the time domain, the Airy phase manifests itself in sudden variations in the amplitude of the temporal signal.

8.3.4 Adiabatic Propagation

The application of the modal expansion to environments with a field of sound speed and variable bathymetries is discussed in detail in [29]; we will refer here only the *adiabatic approximation*, in which the pressure field is represented as

$$P(r, z) \approx \frac{i}{\sqrt{8\pi}} e^{-i\pi/4} \sum_m Z_m(z_s) Z_m(z, r) \frac{e^{i I_m(r)}}{\sqrt{\underline{k}_m(r)\, r}} \tag{8.43}$$

where

$$I_m(r) = \int_0^r \underline{k}_m(s)\, ds,$$

that is, the modes are calculated progressively over distance and the field phase is calculated integrating the horizontal wavenumbers over distance.

8.3.5 Modes and the Reflection Coefficient

The reflection coefficient from ray tracing can find a correspondence with the boundary conditions imposed on the modes; such conditions can be written in the general case as

$$\alpha_1 Z(0) + \beta_1 \left(\frac{dZ}{dz} \right)_{z=0} = 0$$

for the surface and

$$\alpha_2 Z(D) + \beta_2 \left(\frac{dZ}{dz} \right)_{z=D} = 0$$

for the bottom, where α_1, α_2, β_1 and β_2 are numerical values, chosen according to the particularities of the considered waveguide. An alternative way of writing the boundary conditions, albeit less convenient, is to write them in the form

$$\left(\frac{Z}{dZ/dz}\right)_{z=0} = -\frac{\beta_1}{\alpha_1} \quad \text{and} \quad \left(\frac{Z}{dZ/dz}\right)_{z=D} = -\frac{\beta_2}{\alpha_2}$$

Without loss of generality let us consider a stationary wave in the proximity of the surface, which can be written as [29]

$$p(z) = e^{i\hat{k}(0)z} + \mathcal{R}e^{-i\hat{k}(0)z}$$

where

$$\hat{k}(0) = \sqrt{\frac{\omega^2}{c^2(0)} - \underline{k}^2}$$

with $\mathcal{R}$ corresponding to the reflection coefficient. Then, the derivative of the mode corresponds to

$$\frac{dp(z)}{dz} = i\hat{k}(0)\left[e^{i\hat{k}(0)z} + \mathcal{R}e^{-i\hat{k}(0)z}\right]$$

We can see that

$$\left(\frac{p}{dp/dz}\right)_{z=0} = -\frac{i}{\hat{k}(0)}\frac{1+\mathcal{R}}{1-\mathcal{R}}$$

Obviously, this ratio can be solved in order to $\mathcal{R}$ for a given pair of values α_1 and β_1. For instance, for $\alpha_1 = 1$ and $\beta_1 = 0$ we can obtain that $\mathcal{R} = -1$, which is nothing more than the reflection coefficient for a press release surface. For the bottom case, the reasoning is similar: let us consider a stationary wave of the form

$$p(z) = e^{-i\hat{k}(D)z} + \mathcal{R}e^{i\hat{k}(D)z}$$

where

$$\hat{k}(D) = \sqrt{\frac{\omega^2}{c^2(D)} - \underline{k}^2}$$

and $\mathcal{R}$ being the bottom reflection coefficient. This time we can see that

$$\left(\frac{p}{dp/dz}\right)_{z=D} = \frac{i}{\hat{k}(D)}\frac{1+\mathcal{R}}{1-\mathcal{R}}$$

Considering now the case $\alpha_2 = 0$ and $\beta_2 = 1$ we can obtain that $\mathcal{R} = 1$, which is nothing more than the reflection coefficient for a rigid bottom.

For an arbitrary type of boundary, one can consider that the "modal" reflection coefficient is defined for a discrete set of grazing angles, defined by the relationship

$$\tan \underline{\theta}_m = \frac{\sqrt{\dfrac{\omega^2}{c_{\min}^2} - \underline{k}_m^2}}{\underline{k}_m}$$

8.3.6 The Waveguide Invariant

Transmission loss curves for a Pekeris waveguide (generated by the Python Code Snippet 8.7) are illustrated in Fig. 8.9; the curves are calculated for different frequencies over a common interval of ranges. The curves reveal a surprising pattern of constructive interference, which is linearly proportional to the horizontal distance. This relationship is represented in Underwater Acoustics in the form

$$\frac{\delta\omega}{\delta r} = \beta \frac{\omega}{r} \tag{8.44}$$

or, alternatively,

$$\left(\frac{\omega}{\omega_0}\right) = \left(\frac{r}{r_0}\right)^{\beta} \tag{8.45}$$

where β corresponds to the *waveguide invariant*, defined as [3]

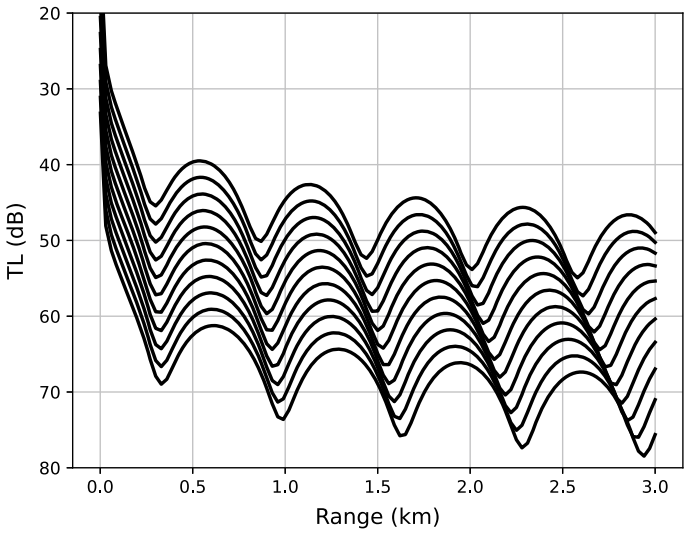


Fig. 8.9 Transmission loss for a Pekeris waveguide, from 21,5 Hz (top) 26 Hz (bottom), separated each by 2 dB. The constructive interference pattern is linearly proportional to the range

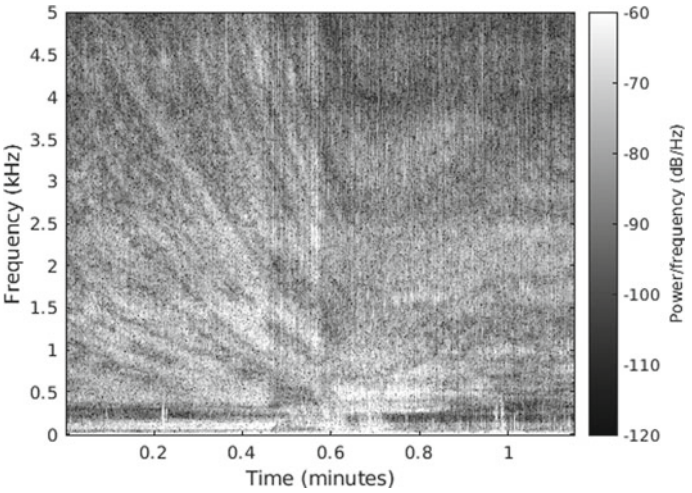


Fig. 8.10 Spectrogram of the propeller noise generated by a surface boat that approaches and moves away from the receiver, revealing striations in noise intensity

$$\frac{1}{\beta} = -\left(\frac{v_p}{v_g}\right)^2 \frac{dv_g}{dv_p} \tag{8.46}$$

For instance, for an isovelocity waveguide we can see that

$$v_p = \frac{\omega}{\underline{k}} = \frac{\omega}{(\omega/c)\cos\underline{\theta}} = \frac{c}{\cos\underline{\theta}} \quad \text{and} \quad v_g = \frac{c^2}{v_p}$$

therefore

$$\frac{1}{\beta} = -\left(\frac{v_p}{c^2/v_p}\right)^2 \frac{d}{dv_p}\left(\frac{c^2}{v_p}\right) = -\left(\frac{v_p^4}{c^4}\right)\left(-\frac{c^2}{v_p^2}\right) = \left(\frac{v_p}{c}\right)^2 = \frac{1}{\cos^2\underline{\theta}}$$

which allows to conclude that $\beta = \cos^2\underline{\theta}$; since intensity values decay rapidly above the critical angle we can consider that $\underline{\theta} \approx 0$, leading to the conclusion that $\beta = 1$.

In the general case, the invariant can be determined using the relationship

$$\frac{\delta\omega}{\delta r} = -\frac{\partial I/\partial r}{\partial I/\partial\omega} \tag{8.47}$$

where I represents the intensity of the acoustic field; the features where the intensity is maximal along frequencies are called *striations* and are typical of the cavitation noise generated by a ship's propellers (see Fig. 8.10); the determination of the waveguide invariant allows, in principle, to determine the approaching and/or distancing of a ship and the corresponding distance using the striations' slope.

Code Snippet 8.7 Invariant revealed in transmission loss curves for a Pekeris waveguide.

```
from scipy.interpolate import interp1d
from numpy import *
from matplotlib.pyplot import *

def kpekeris(c1=None,rho1=None,
             c2=None,rho2=None,
             D=None,freq=None):

    kp = []

    w = 2*pi*freq
    k1 = w/c1
    k2 = w/c2

    nk = 1001
    k = linspace(k1,k2,nk)
    kxk = k*k

    r1 = sqrt( k1*k1 - kxk )
    r2 = sqrt( kxk - k2*k2 )
    s1 = sin( r1*D )
    c1 = cos( r1*D )

    f = s1*r2/rho2 + c1*r1/rho1

    for i in range(nk-1):

        x1 = k[ i ]
        x2 = k[i+1]
        y1 = f[ i ]
        y2 = f[i+1]
        p = y1*y2
        x3  = (x1*y2-x2*y1)/(y2-y1)

        if p < 0:

           kp = append(kp,x3)

    return kp

def pmodes(z=None,Z=None,k=None,
           zs=None,zh=None,rh=None,M=None):

    p = []
    nr = rh.size
    nz = zh.size

    p  = zeros((nz,nr)) + 1j*zeros((nz,nr))
    pm = zeros((nz,nr)) + 1j*zeros((nz,nr))

    e0 = 1j*exp( -1j*pi/4.0 )/sqrt( 8*pi )

    dz = abs( z[1] -z[0] )

    for i in range(M):
```

```
            Zm = Z[:,i]
            interpolator = interp1d(z, Zm)
            Zms = interpolator( zs )
            Zmh = interpolator( zh )
            Zsh = Zms*Zmh
            Rm  = exp( 1j*k[i]*rh )/sqrt( k[i]*rh )
            [RM,ZM] = meshgrid(Rm,Zsh)
            pm  = RM*ZM
            p = p + pm

        p = e0*p
        p = squeeze( p )

        return p

    nfreqs = 11
    fmin   = 21.5
    fmax   = 26.0
    freqs  = linspace(fmin,fmax,nfreqs)
    D      =  100.0
    c1     = 1500.0; rho1 = 1.0
    c2     = 1800.0; rho2 = 1.8
    rmax   = 3000.0
    rh     = linspace(0,rmax,101)
    rh[0]  = 1.0
    rhkm   = rh/1000.0
    z      = linspace(0,D,101); dz = abs( z[1] - z[0] )
    zs     = 36.0
    zh     = array([36.0])
    p0     = 1/( 4*pi )

    for i in range(nfreqs):
        wi = 2*pi*freqs[i]
        kh = kpekeris(c1,rho1,c2,rho2,D,freqs[i])
        M = kh.size
        Z = zeros((101,M))
        kz = sqrt( (wi/c1)**2 - kh*kh )
        for j in range(M):
            Zj = sin( kz[j]*z )
            N = sqrt( dz*sum( Zj*Zj ) )
            Z[:,j] = Zj/N

        p = pmodes(z,Z,kh,zs,zh,rh,M)
        p = p/p0
        tl = -20*log10( abs( p ) ) + i*2.0
        figure(1)
        plot(rhkm,tl,'k',linewidth=2)

    xlabel('Range (km)',fontsize=18)
    ylabel('TL (dB)',fontsize=18)
    ylim(80,20)
    grid(True)

    show()
```

8.3.7 Numerical Calculation of Modes

Beyond the isovelocity and Pekeris waveguides Eq. (8.31) admits analytical solutions in a limited number of cases [60]; for practical purposes, the differential equation is solved numerically, approximating the derivatives as finite differences, and transforming the original problem into a matrix system for which one needs to find a set of eigenvectors and eigenvalues; the substitution can take place in the following way:

- The interval $[0, D]$ is divided into equal segments with indexes $j = 0, 1, 2 \ldots N + 1$, where $h = z_{j+1} - z_j$.
- The first derivative is approximated through the finite difference

$$\frac{dZ}{dz} \approx \frac{Z_{j+1} - Z_j}{h}.$$

- The second derivative is approximated through the finite difference

$$\frac{d^2Z}{dz^2} \approx \frac{Z_{j+1} - 2Z_j + Z_{j-1}}{h^2}.$$

- The boundary conditions at the surface relate Z_0 to Z_1 through the relationship

$$Z_0 = Z_1 \frac{\beta_1/h}{\beta_1/h - \alpha_1}.$$

- The boundary conditions at the bottom relate Z_{N+1} to Z_N through the relationship

$$Z_{N+1} = Z_N \frac{\beta_2/h}{\beta_2/h + \alpha_2}.$$

If we idealize the values Z_1, Z_2, Z_3 arranged into a vector

$$\mathbf{Z} = \begin{bmatrix} Z_1 \\ Z_2 \\ \ldots \\ Z_N \end{bmatrix}$$

then we can represent the system

$$\frac{Z_{j+1}}{h^2} + \left[-\frac{2}{h^2} + \frac{\omega^2}{c^2(z_j)} - \underline{k}^2 \right] Z_j + \frac{Z_{j-1}}{h^2} = 0$$

in a matrix form as

$$\mathbf{AZ} = \underline{k}^2 \mathbf{Z} \tag{8.48}$$

where the matrix $\mathbf{A}$ is tridiagonal:

$$\mathbf{A} = \begin{pmatrix} d_1 & e_2 & & & \\ e_2 & d_2 & e_3 & & \\ & e_3 & d_3 & e_4 & \\ & & & \ddots & \\ & & & e_N & d_N \end{pmatrix}$$

and

$$d_j = \frac{\omega^2}{c^2(z_j)} - \frac{2}{h^2} \quad , \quad e_j = \frac{1}{h^2}$$

with the exceptions

$$d_1 = \frac{1}{h^2}\left[h^2\frac{\omega^2}{c^2(z_1)} + \frac{\beta_1/h}{\beta_1/h - \alpha_1} - 2\right]$$

$$d_N = \frac{1}{h^2}\left[h^2\frac{\omega^2}{c^2(z_N)} + \frac{\beta_2/h}{\beta_2/h + \alpha_2} - 2\right]$$

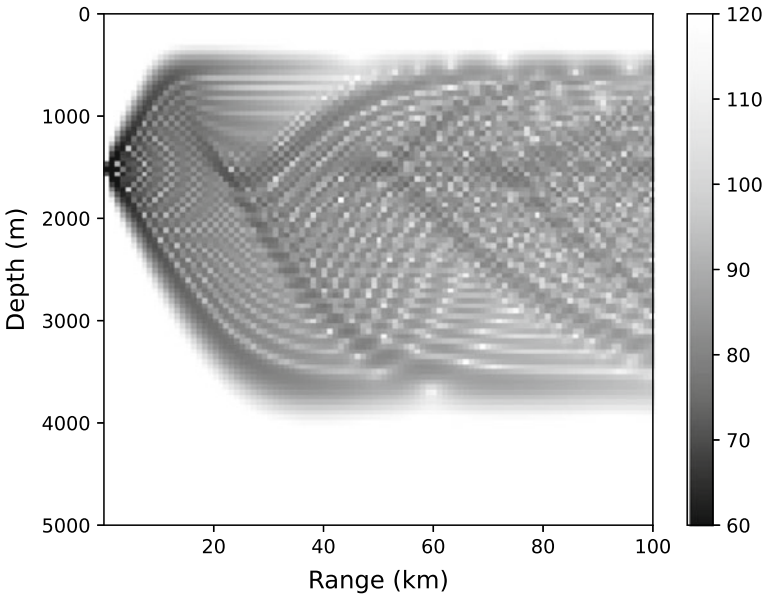


Fig. 8.11 Transmission loss for the Munk profile

which follow from boundary conditions. Equation (8.48) represents a classical problem of calculation of eigenvectors and eigenvalues [55]; each eigenvector corresponds to a mode Z_m calculated at the discretization points, and each eigenvalue corresponds to $\underline{k}_m^2$; trapped modes correspond to eigenvalues $\underline{k}_m^2 > 0$. The numerical implementation of modes calculation is illustrated in Fig. 8.11 through the prediction of transmission loss for the particular case of the Munk profile; the code that calculates the modes, the transmission loss, and generates the figure is shown in the Python Code Snippet 8.8. The interference pattern of the pressure field reveals features of intensity that resemble the ray trajectories shown in Fig. 8.2, with the obvious advantage that the field is defined in coordinates of range and depth and there are no field shadows.

Code Snippet 8.8 Numerical calculation of modes for the Munk profile.

```
from numpy import *
from numpy.linalg import *
from scipy.interpolate import interp1d
from matplotlib.pyplot import *

def modes( z=None, c=None, f=None,
           tbc=None, bbc=None):

    themodes    = []
    wavenumbers = []

    h = abs( z[1] - z[0] )
    hh = h*h

    alpha1 = tbc[0]
    beta1  = tbc[1]
    alpha2 = bbc[0]
    beta2  = bbc[1]

    w = 2*pi*f
    k = w/c
    kk = k*k

    lengthz = z.size
    lengthA = lengthz - 2
    y  = zeros( (lengthz,lengthz) )
    A = zeros( (lengthA,lengthA) )
    Au = zeros( (lengthA,lengthA) )
    Al = zeros( (lengthA,lengthA) )

    ncdiag =  1.0/hh * ones( lengthA-1 )
    cdiag  = -2.0/hh * ones( lengthA   ) + kk[1:-1]

    fill_diagonal( A, cdiag  )

    for i in range(lengthA-1):
        Au[i,i+1] = ncdiag[i]
        Al[i+1,i] = ncdiag[i]
```

```
    A = A + Au + Al
    A[ 0, 0] =  1/hh * ( hh*kk[ 1] + \
    ( -beta1/h )/( alpha1 - beta1/h ) - 2 )
    A[-1,-1] =  1/hh * ( hh*kk[-2] + \
    (  beta2/h )/( alpha2 + beta2/h ) - 2 )

    rankA = linalg.matrix_rank( A )

    if rankA < lengthA:
       print( 'There is no numerical solution \
              for these BCs...' )
       return themodes, wavenumbers

    eigenvalues,modese = linalg.eig( A )

    modes_at_top    = modese[ 0,:]*( -beta1/h )\
                      /( alpha1 - beta1/h )
    modes_at_bottom = modese[-1,:]*(  beta2/h )\
                      /( alpha2 + beta2/h )
    modes = zeros((lengthz,lengthA))
    modes[ 0,:] = modes_at_top
    modes[-1,:] = modes_at_bottom
    modes[1:-1,:] = modese

    indexes =  argsort( eigenvalues )

    eigenvalues = eigenvalues[ indexes ]
    eigenvalues = eigenvalues[::-1]
    modes       = modes[:,indexes]
    modes       = modes[:,::-1]

    poseigen = eigenvalues[eigenvalues > 0]
    wavenumbers = sqrt( poseigen )
    themodes = modes[:,0:poseigen.size]

    for i in range(poseigen.size):
        phi = themodes[:,i]
        if phi[1] < 0:
           phi = -phi
        themodes[:,i] = phi/linalg.norm(phi)

    return themodes, wavenumbers

def pmodes(z=None,Z=None,k=None,
           zs=None,zh=None,rh=None,M=None):

    p = []
    nr = rh.size
    nz = zh.size

    p  = zeros((nz,nr)) + 1j*zeros((nz,nr))
    pm = zeros((nz,nr)) + 1j*zeros((nz,nr))

    e0 = 1j*exp( -1j*pi/4.0 )/sqrt( 8*pi )

    dz = abs( z[1] -z[0] )

    for i in range(M):
```

```
        Zm = Z[:,i]
        interpolator = interp1d(z, Zm)
        Zms = interpolator( zs )
        Zmh = interpolator( zh )
        Zsh = Zms*Zmh
        Rm  = exp( 1j*k[i]*rh )/sqrt( k[i]*rh )
        [RM,ZM] = meshgrid(Rm,Zsh)
        pm  = RM*ZM
        p = p + pm

    p = e0*p
    p = squeeze( p )

    return p

def munk(z,z0,c0):

   B = 1.3e3
   e = 7.37e-3
   eta = 2*( z - z0 )/B
   c   = c0*( 1 + e*( eta + exp( -eta ) - 1 ) )

   return c

D = 5000
nz = 101
z = linspace(0,D,nz)
dz = abs( z[1] - z[0] )
rmaxkm = 100
rmax = rmaxkm*1000
nr = 101
r = linspace(0,rmax,nr)
r[0] = 1.0
rkm = r/1000.0
freq = 50
z0 = 1500.0
c0 = 1480.0
c = munk(z,z0,c0)
p0 = 1.0/( 4*pi )
Z,kh = modes( z, c, freq, [1,0], [1,0])

m = kh.size

for i in range(m):
    Zi = Z[:,i]
    Im = sqrt( dz*sum( Zi*Zi ) )
    Z[:,i] = Zi/Im

p = pmodes(z,Z,kh,z0,z,r,m)
p = p/p0
p = where( p == 0, nan, p )
tl = -20*log10( abs( p ) )

figure(1)
imshow(tl, cmap='Greys_r',
extent=[rkm[0],rkm[-1],z[-1],z[0]],
aspect='auto',origin='upper',vmin=60,vmax=120)
colorbar()
```

```
xlabel('Range (km)',fontsize=18)
ylabel('Depth (m)' ,fontsize=18)

show()
```

8.3.8 *Source Directivity*

As opposed to the discussion of Sect. 8.2.6 the inclusion of source directivity into a modal expansion is not evident, because the expansion itself has not clear parameters related explicitly to specific directions of propagation. In analogy with the weighting of ray amplitudes, the source directivity can be included weighting propagation modes, through the association of each mode with a particular direction of propagation. This can be achieved by calculating the equivalent *modal* angles, defined by the expression

$$\theta_m = \text{acos}\left(\frac{k_m}{k_{\max}}\right)$$

where $k_{\max} = \omega/c_{\min}$ (see Sect. 8.3).

8.4 The WKBJ Approximation

The WKBJ approximation[4] is a phase integral method which lays halfway between ray tracing and normal modes. Let us consider the following solution of Eq. (4.2) [61]:

$$Z(z) = A(z)e^{iS(z)} \,. \tag{8.49}$$

One would have then that

$$\frac{dZ}{dz} = \left(\frac{dA}{dz} + iA\frac{dS}{dz}\right)e^{iS} \,, \tag{8.50}$$

therefore

$$\frac{d^2Z}{dz^2} = \left[\frac{d^2A}{dz^2} - A\left(\frac{dS}{dz}\right)^2 + 2i\frac{dA}{dz}\frac{dS}{dz} + iA\frac{d^2S}{dz^2}\right]e^{iS} \tag{8.51}$$

After substitution into Eq. (4.2), and separating the real and complex parts of the resulting expression, we can arrive at the following system of equations:

[4] After Gregor Wentzel, Henrik A. Kramers, Marcel Louis Brillouin and Harold Jeffreys.

$$2\frac{dA}{dz}\frac{dS}{dz} + A\frac{d^2S}{dz^2} = 0 \tag{8.52}$$

and

$$\frac{d^2A}{dz^2} - A\left(\frac{dS}{dz}\right)^2 + \left(\frac{\omega^2}{c^2} - \underline{k}^2\right)A = 0 \tag{8.53}$$

Solving the first equation of the system, we can obtain that

$$A = \frac{A_0}{\sqrt{dS/dz}}$$

Neglecting now the term d^2A/z^2 in Eq. (8.53) it will follow that

$$\left(\frac{dS}{dz}\right)^2 = \frac{\omega^2}{c^2(z)} - \underline{k}^2$$

which allows to obtain the following expression for $Z(z)$ [29, 59]:

$$Z(z) \approx \frac{1}{\sqrt{\hat{k}(z)}}\left[C_1 e^{iI(z)} + C_2 e^{-iI(z)}\right] \tag{8.54}$$

where

$$\hat{k}^2(z) = \frac{\omega^2}{c^2(z)} - \underline{k}^2 \tag{8.55}$$

and

$$I(z) = \int_{\hat{z}_1}^{z} \hat{k}(s)\, ds \tag{8.56}$$

in which $\hat{k}(\hat{z}_1) = 0$, meaning that $\hat{z}_1$ represents a turning depth, if that is the case; as presented by Eq. (8.54) there seems to be no restrictions regarding the choice of the function $Z(z)$, since arbitrary values of $\underline{k}$ will correspond to different depths $\hat{z}$; let us admit additionally that the sound speed profile is such that allows to define two turning depths for each value of $\underline{k}$, thus defining $Z(z)$ only in the interval $z \in [\hat{z}_1, \hat{z}_2]$; outside this interval $\hat{k}^2(z) < 0$ and we can replace $\hat{k}(z)$ with $i\hat{k}(z)$, giving $Z(z)$ an exponential decay. In order to obtain a set of "modal" functions we should restrict the choice of the $\hat{k}$ only to those satisfying the *Bohr–Sommerfeld condition* [62]:

$$\int_{\hat{z}_1}^{\hat{z}_2} \hat{k}(z)\, dz = \left(m + \frac{1}{2}\right)\pi$$

where $m = 1, 2, 3, \ldots$, meaning that we are selecting only mode oscillations with an integer number of periods between the turning depths. Once the modes had been selected in this way we can find the constants C_1 and C_2 that allow the mode to transit from an oscillatory behaviour to an exponential decay. The reasoning for waveguides without turning depths is similar and is discussed in detail in [60], which presents also the expressions needed to calculate the constants C_1 and C_2.

8.5 Fast-Field Models

An integral representation of the pressure field as a solution for Eq. (4.2) is given by

$$P(r, z) = \int_0^\infty \mathcal{G}(\underline{k}, z)\ \underline{k}\ J_0(\underline{k}r)\ d\underline{k} \tag{8.57}$$

where $J_0(r)$ represents the Bessel function of zeroth order [55]. Since the pressure field is obtained by integration along wavenumbers the solution is known as a *Fast Field*. Substitution of Eq. (8.57) into Eq. (4.2) allows to obtain the following differential equation for $\mathcal{G}(\underline{k}, z)$:

$$\frac{d^2\mathcal{G}}{dz^2} + \left(\frac{\omega^2}{c^2} - \underline{k}^2\right)\mathcal{G} = \delta(z - z_s) \tag{8.58}$$

which shows that $\mathcal{G}$ is none other than a *Green function*, to be calculated for *any* value of $\underline{k}$. In this way, solving Eq. (8.58) in an interval of horizontal wavenumbers allows, through the integral of Eq. (8.57), to calculate the acoustic field along distance.

Let us illustrate the calculation of the Green function for the isovelocity problem, discussed in Sect. 8.3.1, which can be now formulated as the calculation of the function $G(z, z_s)$, for which is valid that

$$\frac{d^2\mathcal{G}}{dz^2} + \left(\frac{\omega^2}{c^2} - \underline{k}^2\right)\mathcal{G} = \delta(z - z_s)$$

with boundary conditions

$$\mathcal{G}(0, z_s) = 0 \quad \text{and} \quad \left(\frac{d\mathcal{G}}{dz}\right)_{z=D} = 0$$

In the interval $z \in [0, z_s]$ the solution can be written in the form

$$\mathcal{G}(z, z_s) = A_1 \sin\left(\hat{k}z\right) + A_2 \cos\left(\hat{k}z\right)$$

where

$$\hat{k} = \sqrt{\frac{\omega^2}{c^2} - \underline{k}^2}$$

The condition at $z = 0$ implies that $A_2 = 0$. For the interval $z \in [z_s, D]$ the solution can be represented as

$$\mathcal{G}(z, z_s) = B_1 \sin\left[\hat{k}(z - D)\right] + B_2 \cos\left[\hat{k}(z - D)\right]$$

By the condition at $z = D$ we can conclude that $B_1 = 0$; additionally, the Green function should be continuous at $z = z_s$, which implies that

$$A_1 \sin\left(\hat{k} z_s\right) = B_2 \cos\left[\hat{k}(z_s - D)\right]$$

On another side the derivative of the Green function should be discontinuous at $z = z_s$, and therefore [55]

$$\frac{d\mathcal{G}}{dz}(z_s + 0, z_s) - \frac{d\mathcal{G}}{dz}(z_s - 0, z_s) = 1$$

which in this particular case corresponds to

$$-\hat{k} B_2 \sin\left[\hat{k}(z_s - D)\right] - \hat{k} A_1 \cos\left(\hat{k} z_s\right) = 1$$

leading to

$$A_1 = \frac{-\underline{C}_2}{\hat{k}\left(\underline{C}_1 \underline{C}_2 + \underline{S}_1 \underline{S}_2\right)}$$

and

$$B_2 = \frac{-\underline{S}_1}{\hat{k}\left(\underline{C}_1 \underline{C}_2 + \underline{S}_1 \underline{S}_2\right)}$$

where

$$\underline{S}_1 = \sin\left(\hat{k} z_s\right) \quad , \quad \underline{S}_2 = \sin\left[\hat{k}(z_s - D)\right]$$

$$\underline{C}_1 = \cos\left(\hat{k} z_s\right) \quad , \quad \underline{C}_2 = \cos\left[\hat{k}(z_s - D)\right]$$

Based on the relationship $\cos(\alpha - \beta) = \cos\alpha \cos\beta + \sin\alpha \sin\beta$ it can be obtained that

$$\underline{C}_1 \underline{C}_2 + \underline{S}_1 \underline{S}_2 = \cos\left(\hat{k} D\right)$$

which allows to write finally the Green function as

$$\mathcal{G}(z, z_s) = \begin{cases} -\cos\left[\hat{k}\,(z_s - D)\right]\sin\left(\hat{k}z\right)/W \text{ for } z < z_s \\ -\sin\left(\hat{k}z_s\right)\cos\left[\hat{k}\,(z - D)\right]/W \text{ for } z > z_s \end{cases}$$

where

$$W = \hat{k}\cos\left(\hat{k}D\right)$$

represents the *Wronskian* of the independent solutions that allow to calculate $\mathcal{G}(z, z_s)$. In the general case, the wavenumbers in which the Wronskian is zero define the eigenvalues associated to the modes of the waveguide [29]; such eigenvalues allow to approximate the fast-field solution as a sum or residuals, that is, they establish a connection between the modal expansion and the fast field. In the particular case here discussed the condition $\cos\left(\hat{k}D\right) = 0$ allows to deduce one more time the eigenvalues of the modal solutions discussed in Sect. 8.3.1.

From the numerical point of view, the calculation of the Green function is equivalent to a problem of Linear Algebra

$$\mathbf{BG}(k) = \boldsymbol{\delta}$$

where [3]

$$\mathbf{B} = \mathbf{A} - k^2\mathbf{I}$$

$$\mathbf{A} = \begin{pmatrix} d_1 & e_2 & & & \\ e_2 & d_2 & e_3 & & \\ & e_3 & d_3 & e_4 & \\ & & & \ddots & \\ & & & e_N & d_N \end{pmatrix}$$

with

$$d_j = \frac{\omega^2}{c^2(z_j)} - \frac{2}{h^2} \quad , \quad e_j = \frac{1}{h^2}$$

and $\boldsymbol{\delta}$ representing the numerical equivalent of a Dirac function, that is, corresponding to a vector with all components set to zero except at the index that corresponds to the source depth, where it is equal to 1; the elements of the matrix $\mathbf{A}$ are valid for the boundary conditions with $\alpha_1 = \alpha_2 = 1$ and $\beta_1 = \beta_2 = 0$. Each value of k allows to calculate a vector $\mathbf{G}(k)$, which is the equivalent of $\mathcal{G}(k, z)$ and the pressure field can be calculated based on the approximation

$$P(r, z) \approx \frac{e^{i\pi/4}}{\sqrt{2\pi r}} \int_{k_{\min}}^{k_{\max}} \mathcal{G}(k, z)\,\sqrt{k}\,e^{ikr}dk$$

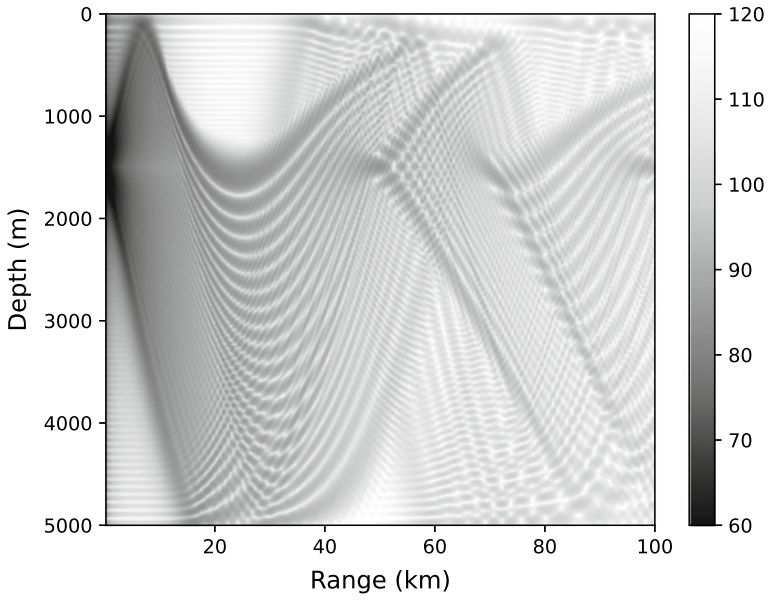


Fig. 8.12 Fast-field transmission loss for the Munk profile

The implementation of this numerical approximation for the particular case of the Munk profile is illustrated in Fig. 8.12, which is generated by the Python Code Snippet 8.9; in order to preserve the stability of wavenumber integration, the wavenumbers are slightly displaced into the complex plane. The interference pattern shown in Fig. 8.12 reveals many similarities with the one shown in Fig. 8.11, but with more detail, besides exhibiting a larger angular aperture of the propagating field when compared with the modal prediction. Again features of interference resemble some of the ray trajectories shown in Fig. 8.2.

The extension of the fast-field method to waveguides with elastic bottoms and associated discretization issues of the wavenumbers is discussed in detail in [26]; the adaptation to a medium with a velocity field and bathymetries that depend on the horizontal distance is a subject of discussion in [3].

Code Snippet 8.9 Fast-field calculation of transmission loss.

```
from numpy import *
from numpy.linalg import *
from matplotlib.pyplot import *

def ffpA( z=None, c=None, freq=None, k=None):

    A = []
```

```
    h = abs( z[1] - z[0] )
    hh = h*h

    w = 2*pi*freq
    ww = w*w
    cc = c*c
    lengthA = z.size
    A  = zeros( (lengthA,lengthA) )

    cdiag  = -2.0/hh + ww/cc

    fill_diagonal( A,  cdiag  )

    for i in range(lengthA-1):
        A[i,i+1] = 1.0/hh
        A[i+1,i] = 1.0/hh

    A = A - k*k*eye( lengthA )

    return A

def munk(z,z0,c0):

   B = 1.3e3
   e = 7.37e-3
   eta = 2*( z - z0 )/B
   c   = c0*( 1 + e*( eta + exp( -eta ) - 1 ) )

   return c

D = 5000
nz = 500
z = linspace(0,D,nz); dz = z[1] - z[0]
freq = 50; w = 2*pi*freq
z0 = 1500.0
zs = z0
c0 = 1480.0
c = munk(z,z0,c0)
k = w/c
kmin = min( k )
kmax = max( k )
nk = 501
k = linspace(kmin,kmax,nk)
dk = k[1] - k[0]
k = k - 1j*dk
nr     = 501
rmaxkm = 100
rmax = rmaxkm*1000
dr = rmax/nr
r = linspace(dr,rmax,nr); rkm = r/1000.0
p = zeros((nz,nk)) + 1j*zeros((nz,nk))
d = zeros( nz )
index = int( zs/dz )
d[ index ] = 1.0 # Dirac

G = zeros((nz,nk)) + 1j*zeros((nz,nk))

for i in range(nk):
```

```
    A = ffpA( z, c, freq, k[i])
    G[:,i] = linalg.solve(A,d)

for i in range(nr):
     kr = k*r[i]
     for j in range(nz):
         I = G[j,:]*sqrt( k )*exp( 1j*kr )
         p[j,i] = dk*sum( I )/r[i]

p = 4*pi*p*exp( 1j*pi/4 )/sqrt( 2*pi )
p = where( p == 0, nan, p )
tl = -20*log10( abs( p ) )

figure(1)
imshow(tl, cmap='Greys_r',
extent=[rkm[0],rkm[-1],z[-1],z[0]],
aspect='auto',origin='upper',vmin=60,vmax=120)
colorbar()
xlabel('Range (km)',fontsize=18)
ylabel('Depth (m)',fontsize=18)

show()
```

8.6 The Parabolic Equation

Let us consider now the following representation of the pressure field as a solution of Eq. (4.2):

$$P(r,z) = U(z,r)H_0^{(1)}(k_0 r) \tag{8.59}$$

where $U(z,r)$ is a function that weakly depends on range [63] and k_0 is a "reference" wavenumber. Substituting into Eq. (4.2) one can obtain the following differential equation for U:

$$\frac{\partial^2 U}{\partial r^2} + \frac{\partial^2 U}{\partial z^2} + \left(\frac{1}{r} + \frac{2}{V}\frac{\partial V}{\partial r}\right)\frac{\partial U}{\partial r} + k_0^2\left(n^2 - 1\right)U = 0 \tag{8.60}$$

where

$$n = \frac{c_0}{c(r,z)}$$

with c_0 representing a reference velocity and $V = H_0^{(1)}(k_0 r)$. The asymptotical behaviour of $H_0^{(1)}(k_0 r)$ for large distances allows to rewrite Eq. (8.60) in the form:

$$\frac{\partial^2 U}{\partial r^2} + \frac{\partial^2 U}{\partial z^2} + 2ik_0\frac{\partial U}{\partial r} + k_0^2\left(n^2 - 1\right)U = 0 \tag{8.61}$$

Admitting that $\partial/\partial r$ and $\partial/\partial z$ mutually commute we can now factorize Eq. (8.61) in the form:

$$\left(\frac{\partial}{\partial r} + ik_0 - ik_0\sqrt{1+L}\right)\left(\frac{\partial}{\partial r} + ik_0 + ik_0\sqrt{1+L}\right)U = 0 \tag{8.62}$$

where

$$L = \left(n^2 - 1\right) + \frac{1}{k_0^2}\frac{\partial^2}{\partial z^2} \tag{8.63}$$

Neglecting divergent waves we can reduce this equation to

$$\left(\frac{\partial}{\partial r} + ik_0 - ik_0\sqrt{1+L}\right)U = 0 \tag{8.64}$$

This expression can be approximated in different ways, giving rise to different types of *parabolic equations*, namely:

- Rational approximation

$$\sqrt{1+L} \approx 1 + \frac{1}{2}L$$

  leading to *Tappert's parabolic equation* [6]:

$$\frac{\partial U}{\partial r} = \frac{i}{2k_0}\frac{\partial^2 U}{\partial z^2} + \frac{ik_0}{2}\left(n^2 - 1\right)U \tag{8.65}$$

  which is valid for a source aperture of 10°.
- Polynomial approximation

$$\sqrt{1+L} \approx \frac{1 + 3L/4}{1 + L/4}$$

  which leads to *Claerbout's equation* [3, 64]:

$$\left[1 + \frac{1}{4}\left(n^2 - 1\right)\right]\frac{\partial U}{\partial r} + \left(\frac{1}{2k_0}\right)^2 \frac{\partial^3 U}{\partial z^2 \partial r} =$$

$$= \frac{i}{2k_0}\frac{\partial^2 U}{\partial z^2} + \frac{ik_0}{2}\left(n^2 - 1\right)U \tag{8.66}$$

  and can be solved numerically with the Crank–Nicolson method. Claerbout's equation is valid for a source aperture of 40°.
- Padé polynomial approximation

$$\sqrt{1+L} \approx 1 + \sum_{j=1}^{M} \frac{a_{2j-1,M} L}{1 + a_{2j,M} L}$$

where the $a_{2j,M}$ are known as *Padé coefficients*. The corresponding parabolic equation is given by [63]

$$\frac{\partial U}{\partial r} = ik_0 \left(1 + \sum_{j=1}^{M} \frac{a_{2j-1,M} L}{1 + a_{2j,M} L} \right) U \tag{8.67}$$

where

$$a_{2j-1,M} = \frac{2}{2M+1} \sin^2 \left(\frac{j\pi}{2M+1} \right)$$

and

$$a_{2j,M} = \cos^2 \left(\frac{j\pi}{2M+1} \right) \tag{8.68}$$

Equation (8.67) can be solved numerically with Galerkin's method or with the Crank–Nicolson method. Increasing the number of Padé coefficients allows to increase the angular validity of the operator $\sqrt{1+L}$, but the improvement in accuracy implies an increase in the time of calculations.

The fundamental goal of parabolic equation models is to include arbitrary variations in sound speed and/or bathymetry, which in the case of normal modes and the fast-field method require adaptations of the original solution. In general, solving any parabolic equation requires a field *starter*, that is, an initial condition in order to proceed with the progressive calculation of the acoustic field. The starter can be a solution calculated with normal modes or an analytic solution representative of a Dirac; a common choice is to select the starter given by [3]

$$U(0, z) = \sqrt{k_0} e^{-k_0^2/2(z-z_s)^2}$$

From another side, since in all parabolic equations the field is calculated progressively over range, the accuracy of the results depends significantly on the discretization of the medium, both in range and depth. Therefore, computational calculations are sometimes demanding. As indicated previously parabolic equation models do not take into account the signal returned to the source.

From the numerical point of view, the solution of a parabolic equation is equivalent to a matrix problem of the type

$$\mathbf{C}\mathbf{U}_{i+1} = \mathbf{B}\mathbf{U}_i$$

where $\mathbf{U}_1$ corresponds to the discretized version of the starter and the matrices $\mathbf{C}$ and $\mathbf{B}$ are calculated depending on the type of parabolic approximation. For Tappert's

equation, for instance, **B** and **C** are tridiagonal matrices, whose elements can be written as [3]

$$B_{ij} = \frac{2ik_0}{\Delta r} - \frac{1}{2}A_{ij} \quad \text{and} \quad C_{ij} = \frac{2ik_0}{\Delta r} + \frac{1}{2}A_{ij}$$

where

$$\mathbf{A} = \begin{pmatrix} d_1 & e_2 & & & \\ e_2 & d_2 & e_3 & & \\ & e_3 & d_3 & e_4 & \\ & & & \ddots & \\ & & & e_N & d_N \end{pmatrix}$$

with

$$d_j = k_0^2\left[n^2(z_j) - 1\right] - \frac{2}{h^2} \quad \text{and} \quad e_j = \frac{1}{h^2}$$

These expressions are valid for boundary conditions with $\alpha_1 = \alpha_2 = 1$ and $\beta_1 = \beta_2 = 0$. The calculation of transmission loss with Tappert's equation is illustrated in Fig. 8.13, which is generated by the Python Code Snippet 8.10. As in the case of Fig. 8.11 and of Fig. 8.12, the pressure field reveals a constructive interference akin to propagation rays, but the angular limitation of Tappert's equation manifests itself as a limited field aperture. The level of field detail can be improved with a proper choice of discretization in range and depth.

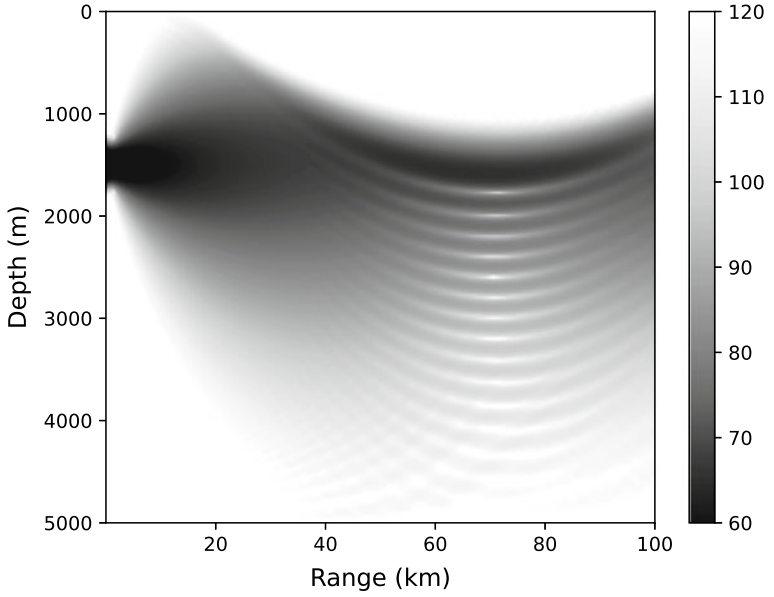


Fig. 8.13 Transmission loss for the Munk profile using Tappert's parabolic equation

Code Snippet 8.10 Transmission loss calculations with Tappert's parabolic equation.

```
from numpy import *
from numpy.linalg import *
from matplotlib.pyplot import *

def peBC( z=None, n=None, k0=None, dr=None):

    B = []
    C = []

    h = abs( z[1] - z[0] )
    hh = h*h

    lengthA = z.size
    A  = zeros( (lengthA,lengthA) )
    B  = zeros( (lengthA,lengthA) ) + \
      1j*zeros( (lengthA,lengthA) )
    C  = zeros( (lengthA,lengthA) ) + \
      1j*zeros( (lengthA,lengthA) )

    cdiag  = -2.0/hh + k0*k0*( n*n - 1 )

    fill_diagonal( A,  cdiag  )
    fill_diagonal( B, -cdiag*0.5 + 2*1j*k0/dr  )
    fill_diagonal( C,  cdiag*0.5 + 2*1j*k0/dr  )

    for i in range(lengthA-1):
        A[i,i+1] = 1.0/hh
        A[i+1,i] = 1.0/hh
        B[i,i+1] = 2*1j*k0/dr - 0.5/hh
        B[i+1,i] = 2*1j*k0/dr - 0.5/hh
        C[i,i+1] = 2*1j*k0/dr + 0.5/hh
        C[i+1,i] = 2*1j*k0/dr + 0.5/hh

    return B,C

def munk(z,z0,c0):

   B = 1.3e3
   e = 7.37e-3
   eta = 2*( z - z0 )/B
   c   = c0*( 1 + e*( eta + exp( -eta ) - 1 ) )

   return c

D = 5000
nz = 201
z = linspace(0,D,nz)
rmaxkm = 100; rmax = rmaxkm*1000
nr = 501
r = linspace(0,rmax,nr)
dr = r[1] - r[0]
r[0] = 1.0
rkm = r/1000.0
freq = 50; w = 2*pi*freq
z0 = 1500.0
```

```
c0 = 1480.0
c = munk(z,z0,c0)
k0 = w/c0
n  = c0/c
p = zeros((nz,nr)) + 1j*zeros((nz,nr))
U = zeros((nz,nr)) + 1j*zeros((nz,nr))
ks = k0/15
U[:,0] = sqrt( ks )*exp( -0.5*ks*ks*( z - z0 )**2 )

B,C = peBC(z, n, k0, dr)

for i in range(nr-1):
    y = B.dot(U[:,i])
    U[:,i+1] = linalg.solve(C,y)

for i in range(nr):
    H0 = sqrt( 2/(pi*k0*r[i]) )*\
         exp( 1j*( k0*r[i] - pi/4  ) )
    p[:,i] = U[:,i]*H0

p = where( p == 0, nan, p )
tl = -20*log10( abs( p ) )

figure(1)
imshow(tl, cmap='Greys_r',
extent=[rkm[0],rkm[-1],z[-1],z[0]],
aspect='auto',origin='upper',vmin=60,vmax=120)
colorbar()
xlabel('Range (km)',fontsize=18)
ylabel('Depth (m)',fontsize=18)

show()
```

Chapter 9
Scattering and Reverberation

9.1 Introduction

The concept of reverberation was introduced in Sect. 6.10 and is going to be discussed in more detail in this chapter. It should be taken into account that in the general case:

- Reverberation can be considered as a particular case of *scattering*, in which the energy of a wave incident on a boundary is redirected in *all* directions, besides the specular one. In a certain sense, scattering is like a "generalized" reflection, within and outside the plane of incidence, but with most of the wave energy being reflected at the boundary along the specular direction; in this context, reverberation includes only the scattering that allows the "return" of energy to the source (*backscattering*); the factor that makes scattering possible is the boundary *roughness*.
- Different boundaries can be characterized with different types of roughness, and both surface and bottom are reverberating boundaries, therefore there are different models for surface and bottom scattering.
- Roughness is always microscopic in comparison with the typical scales of propagation range and depth, which makes scattering to be strongly dependent on signal's frequency and in fact increases as frequency increases. This complex dependence makes impractical the development of theoretical models; instead (as with equations for the calculation of sound speed) scattering models are semi-empirical, meaning that they are partially based on theoretical concepts and need to be functionally and parametrically adjusted to a given set of observations.

Before proceeding with the formalism of scattering models it is necessary to define the general geometry of the scattering problem: in a typical scenario the acoustic energy incident on a boundary at a grazing angle $\underline{\theta}$, while the energy is scattered in the directions defined by the scattering and bistatic angles θ_s and ϕ_s, respectively (see Fig. 9.1). Formally, the angles are defined within the intervals

$$\underline{\theta}, \theta_s \in [0, 90^\circ], \phi_s \in [-180^\circ, 180^\circ]$$

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023

O. C. Rodríguez, *Fundamentals of Underwater Acoustics*,
https://doi.org/10.1007/978-3-031-31319-6_9

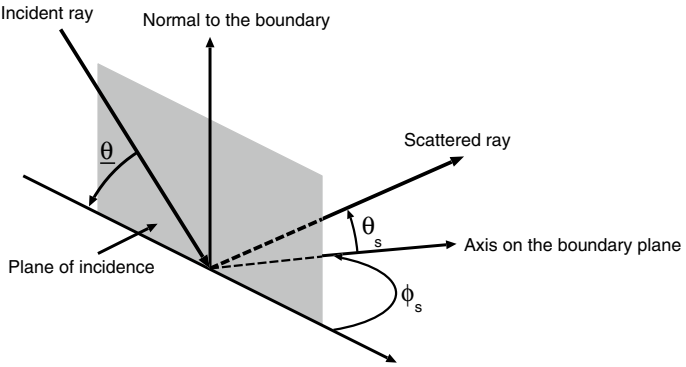


Fig. 9.1 General geometry of the scattering problem

Specular reflection is given by the angles $\theta_s = \underline{\theta}$ and $\phi_s = 0$, while backscattering corresponds to $\theta_s = \underline{\theta}$ and $\phi_s = 180°$. For each value of $\underline{\theta}$, the intensity of scattering should be maximal in the region that corresponds to $\theta_s \approx \underline{\theta}$ and $\phi_s \approx 0$; for a fixed value of $\underline{\theta}$ the intensity of scattering can be visualized on a polar diagram, with θ_s in the role of radial coordinate and ϕ_s in the role of angular coordinate; such a diagram should exhibit a "radial" symmetry, that is, scattering values for ϕ_s and $-\phi_s$ should be the same; additionally, hard bottoms should scatter energy with more intensity than soft bottoms.

In the general case, scattering calculations resort to the scattering *cross section* σ, which can be idealized as a sort of squared reflection coefficient; however, unlike the reflection coefficient, σ allows to take into account only the intensity of the scattered wave, meaning that it carries no phase information. The equivalent of σ within the context of the SONAR equation corresponds to the *scattering strength* SS[1], which is defined as

$$\mathrm{SS} = 10 \log \sigma \tag{9.1}$$

in which [SS] = dB. The relationship between the scattering strength and the reverberation level is given by

$$\mathrm{RL} = \mathrm{SL} - 2\mathrm{TL} + \mathrm{SS} + \underline{\mathrm{RL}} \tag{9.2}$$

where $\underline{\mathrm{RL}}$ represents the boundary reverberation. The remaining of this chapter is organized as follows: Sect. 9.2 will introduce the simplest model of scattering, called *Lambert's model*; surface scattering based on Kirchhoff scatter theory will be addressed in Sect. 9.3; surface backscattering is presented in Sect. 9.4; bottom scattering will be presented in Sect. 9.5; boundary reverberation, which depends on the duration of the transmitted pulse, will be discussed in Sect. 9.7.

[1] Not to be confused with the *Sea State*.

9.2 Lambert's Model

Lambert's law represents a simplified model of scattering, in which the scattering cross section is written in the form [33]:

$$\sigma = A\varepsilon_0 \frac{\cos\theta \cos\theta_s}{\pi} \tag{9.3}$$

where A and ε_0 are parameters to be determined from experimental data. Equation (9.3) has obvious limitations, because

- it is frequency independent;
- it is independent of ϕ_s;
- its application is limited to the availability of experimental data.

9.3 Surface Scattering

From an acoustic point of view, surface roughness can be characterized through the Rayleigh roughness parameter, which is given by [65]

$$R = 2k\sigma \sin\theta = \frac{4\pi\sigma \sin\theta}{\lambda} \tag{9.4}$$

where σ represents the RMS surface-wave height (crest to trough), θ is the grazing angle, λ is the acoustic wavelength and k is the wavenumber ($k = 2\pi/\lambda$). In the case when $R \ll 1$ the surface acts mostly as a reflector and produces a coherent reflection at the specular angle, which is equal to the angle of incidence; when $R \gg 1$ the surface acts as a scatterer, scattering energy in all directions. Reflectivity from a rough surface can be characterized through the relationship [3]

$$\mathcal{R}'(\theta) = \mathcal{R}(\theta) \exp\left(-\frac{R^2}{2}\right) \tag{9.5}$$

where $\mathcal{R}'$ and $\mathcal{R}$ correspond to the new and old reflection coefficients, respectively, with $\mathcal{R}'$ being reduced because of scattering at the rough surface. Surface roughness can be wind-driven and/or modulated by swells; an overview of models for sea surface roughness is presented in Appendix A.

9.4 Surface Backscattering

At the surface there are two fundamental factors responsible for backscattering, namely, the roughness induced by the sea state and the existence of air bubbles in the proximity of the surface. In the interval of small to "moderate" angles, backscattering

is dominated by air bubbles and in the rest of the interval it is dominated by the sea roughness. Each of these factors corresponds to a cross section, which for the interval from 10 to 100 kHz can be modelled in the following way [66]:

- Air bubbles: for a frequency f in kHz and for a wind speed U in m/s the cross section can be calculated using the following expression:

$$\sigma_b = \frac{\beta_v}{4\pi} \frac{\delta_r}{\delta} \left[\frac{1 + 8\beta e^{-2\beta} - e^{-4\beta}}{2\beta} \right] \tag{9.6}$$

where $\beta = \beta_v / \sin \underline{\theta}$, $\delta_r = 0.0136$ (the resonant coefficient), $\delta = 2.255 \times 10^{-2} \sqrt[3]{f}$

$$\beta_v = 10^{-5.2577+0.4701U} \, (f/25)^{0.85}$$

for $U \leq 11$ m/s and

$$\beta_v = \beta_v(U = 11 \text{ m/s}) \, (U/11)^{3/5}$$

for $U \geq 11$ m/s. For a high density of bubbles ($\beta \gg 1$) the cross section becomes independent of wind speed and corresponds to

$$\sigma_b = \frac{\sin \underline{\theta}}{8\pi} \frac{\delta_r}{\delta}$$

while for low densities ($\beta \ll 1$) one would have that

$$\sigma_b = \frac{3\beta}{2\pi} \frac{\delta_r}{\delta}$$

- Surface roughness: The corresponding cross section is composed of two terms

$$\sigma_r = \sigma_f + \sigma_{sc} \tag{9.7}$$

where σ_f represents the cross section induced by the surface facets and σ_{sc} is the cross section induced by small-scale roughness. Each of the cross sections is calculated as follows:

  - Small-scale roughness is given by the expression

$$\sigma_{sc} = \begin{cases} A(U) \tan^4(\underline{\theta}) & \text{when } \underline{\theta} \leq 85^\circ \\ 0 & \text{when } \underline{\theta} > 85^\circ \end{cases} \tag{9.8}$$

in which

$$A(U) = 1.3 \times 10^{-5} U^2$$

  - Facet roughness is given by the expression

$$\sigma_f = \frac{\sec^4 \theta}{4\pi s^2} \exp\left[-\frac{\tan^2 \theta}{s^2}\right] \tag{9.9}$$

where the mean facet slope s is given by

$$s^2 = \begin{cases} 4.6 \times 10^{-3} \ln\left(2, 1U^2\right) & \text{when } U \geq 1 \text{ m/s} \\ 0.0034 & \text{when } U < 1 \text{ m/s} \end{cases} \tag{9.10}$$

In order to deal with the transition of σ_{sc} at $\underline{\theta} = 85°$, it is used in the following interpolation formula:

$$\sigma_{ri} = f(x)\sigma_f + [1 - f(x)]\,\sigma_{sc}$$

where

$$f(x) = \frac{1}{1 + e^x}$$

and

$$x = 0.524\left(\underline{\theta}_f - \underline{\theta}\right)$$

with $\underline{\theta}_f$ and $\underline{\theta}$ given in degrees; the parameter $\underline{\theta}_f$ corresponds to the angle for which $\sigma_f(\underline{\theta}_f)$ is 15 dB below $\sigma_f(90°)$. The final value of σ_r is calculated with the expression

$$\sigma_r = \sigma_{ri} 10^{\mathrm{SBL}/10} \tag{9.11}$$

in which

$$\mathrm{SBL} = \begin{cases} \dfrac{1.26 \times 10^{-3}}{\sin \underline{\theta}} U^{1.57} f^{0.85} & \text{when } U \geq 6 \text{ m/s} \\ \mathrm{SBL}(U = 6 \text{ m/s}) e^{1.2(U-6)} & \text{when } U < 6 \text{ m/s} \end{cases} \tag{9.12}$$

The total cross section corresponds then to

$$\sigma = \sigma_r + \sigma_b \tag{9.13}$$

Figure 9.2, generated by the Code Snippet 9.1, illustrates the behaviour of surface backscattering for a frequency of 25 kHz and different wind speeds; the figure reveals that the backscattering curves differ significantly from each other when the wind speed is inferior to 8 m/s.

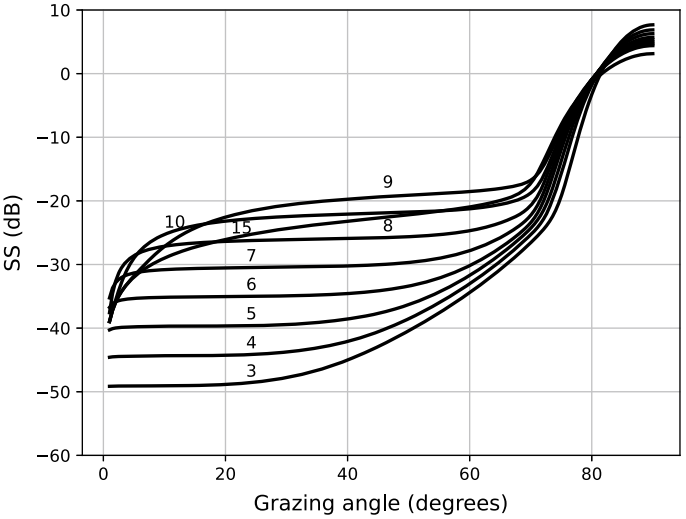


Fig. 9.2 Backscattering strength for a frequency of 25 kHz and different values of wind speed, in m/s

Code Snippet 9.1 Surface backscattering strength for different values of wind speed

```
from numpy import *
from matplotlib.pyplot import *

def apl_bub_sigma(q=None,freq=None,U=None):

        fkHz = freq/1000.0
        f2 = ( fkHz/25.0 )**0.85
        if U < 11:
           f1 = -5.2577 + 0.4701*U
           bv = f2*( 10.0**f1 )
        else:
           f1 = -5.2577 + 0.4701*11
           bv11 = f2*( 10.0**f1 )
           f3 = (U/11.0)**3.5
           bv = bv11*f3
        b = bv/sin( q )
        dr = 0.0136
        p1 = bv*dr
        d = 2.55e-2*( fkHz**(1.0/3.0) )
        p2 = 4.0*pi*d
        r1 = 1.0 + 8.0*b*exp( -2.0*b ) - exp( -4.0*b )
        r2 = 2.0*b
        r = r1/r2
        s = p1/p2*r

        return s
```

```
def apl_surface_sigma(q = None, freq = None,
                      U = None, qf = None):

       fkHz = freq/1000.0
       A = 1.3e-5*U
       sxs = 0.034
       if ( U >= 1.0 ):
          sxs = 4.6e-3*log( 2.1*U*U )
       p2 = 4.0*pi*sxs
       if ( q*180.0/pi <= 85.0 ):
          ssc = A*( (tan(q))**4.0 )
       else:
          ssc = 0.0
       if (U >= 6 ):
          SBL = 1.26e-3/sin(q)*( U**1.57 )*\
                ( fkHz**0.85 )
       else:
          SBL = 1.26e-3/sin(q)*( 6**1.57 )*\
                ( fkHz**0.85 )*exp( U - 6.0 )
       g = 0.5*pi - q
       secg = 1.0/cos( g )
       p1 = secg**4.0
       x = tan(g)**2.0/sxs
       sf = p1/p2*exp( -x )
       x = 0.524*( qf - q )*180.0/pi
       f = 1.0/( 1.0 + exp(x) )
       sri = f*sf + ( 1.0 - f )*ssc
       sr = sri*10**( -SBL/10.0 )
       return sr

freqkHz = 25; freq = freqkHz*1000

nthetas = 180
thetas  = linspace(1,90,nthetas)
q  = thetas*pi/180.0
qi = 0.5*pi - q
secg = 1.0/cos( qi )
p1 = secg**4.0

U = arange(3.0,11.0) # Wind
U = append(U,15.0)
sU = size( U )

qf    = zeros(sU)
sigma = zeros( (sU,nthetas) )

for i in range(sU):
    sxs = 0.034
    Ui = U[i]
    if ( Ui >= 1.0 ):
       sxs = 4.6e-3*log( 2.1*Ui*Ui )
    p2 = 4.0*pi*sxs
    sf90 = 1.0/p2
    x = tan(qi)**2.0/sxs
    sf = p1/p2*exp( -x )
    threshold = sf90/( 10**1.5 )
    index = argmax( sf >= threshold )
    qf[i] = q[index]
```

```
        for j in range(nthetas):
            sb = apl_bub_sigma(q[j],freq,U[i])
            sr = apl_surface_sigma(q[j],freq,U[i],qf[i])
            sigma[i,j] = sr + sb

SS = 10*log10( sigma )

figure(1)
for i in range(sU):
    plot(thetas,SS[i,],'k',linewidth=2)
    if i < 5:
        text(thetas[45],SS[i,45]+1,str(int(U[i])))
    elif i == 5:
        text(thetas[90],SS[i,90]+1,str(int(U[i])))
    elif i == 6:
        text(thetas[90],SS[i,90]+4,str(int(U[i])))
    elif i == 7:
        text(thetas[18],SS[i,18]+3,str(int(U[i])))
    else:
        text(thetas[40],SS[i,40]+0.75,str(int(U[i])))

xlabel('Grazing angle (degrees)',fontsize=18)
ylabel('SS (dB)',fontsize=18)
ylim(-60,10)
grid(True)

show()
```

9.5 Bottom Scattering

The starting point for the calculation of bottom cross section is given by the definition of bottom properties, whose units need to be properly adapted to the scattering case; such units are indicated in Table 9.1. Bottom properties are combined with water density ρ_w and water sound speed c_w, above the bottom, to calculate the following set of parameters:

$$\begin{gathered} a_\rho = \rho_b/\rho_w \ , \\ \nu_{p,s} = c_{p,s}/c_w \ , \qquad \delta_{p,s} = \alpha_{p,s}\nu_{p,s} \ln 10/(40\pi) \\ \tilde{c}_{p,s} = c_p(1 - i\tilde{\alpha_{p,s}})/(1 + \tilde{\alpha}_{p,s}^2) \ , \quad \tilde{\alpha}_{p,s} = \alpha_{p,s}/(40\pi \log e) \end{gathered} \tag{9.14}$$

To this set of parameters one need to add the parameters of bottom roughness, which are indicated in Table 9.2; specific values of these parameters for some bottom types are shown in Tables 9.3 and 9.4 [67].

The model that is being presented here is valid for a frequency interval between 10 and 100 KHz and it is discussed in detail in [66]; its formalism is rather elaborate and, therefore, we will describe it in broad terms and will provide its numerical

Table 9.1 Bottom parameters

Parameter	Symbol	Units
Density	ρ_b	(g/cm^3)
Compressional speed	c_p	m/s
Shear speed	c_s	m/s
Compressional attenuation	α_p	dB/λ
Shear attenuation	α_s	dB/λ

Table 9.2 Input parameters for the bottom scattering model

Parameter	Symbol	Comment
Density ratio	a_ρ	Dimensionless
Compressional ratio	ν_p	Idem
Shear ratio	ν_s	Idem
Compressional loss	δ_p	Idem
Shear loss	δ_s	Idem
Compressional parameter	$\tilde{c}_p$	Complex, velocity
Shear parameter	$\tilde{c}_s$	Complex, velocity
Roughness spectral exponent	γ_2	$2 < \gamma_2 < 4$
Volume spectral exponent	γ_3	$3 < \gamma_3 < 5$
Roughness spectral level	w_2	$[w_2] = \mathrm{m}^{4-\gamma_2}$
Roughness spectral level	w_3	$[w_3] = \mathrm{m}^{3-\gamma_3}$

Table 9.3 Scattering parameters for different types of bottoms (c_w = 1500 m/s, ρ_w = 1 g/cm^3)

Bottom	a_ρ	ν_p	δ_p	ν_s	δ_s
Mud	1.149	0.9873	0.00386	0.002	1
Fine sand	1.451	1.1072	0.01602	0.002	1
Medium sand	1.845	1.1782	0.01624	0.002	1
Coarse sand	2.231	1.2503	0.01638	0.134	0.075
Fine gravel	2.492	1.337	0.01705	0.156	0.2
Gravel	2.5	1.8	0.01374	1.01	0.1
Rock	2.5	2.3	0.00174	1.3	0.085
Basalt	2.5	2.3	0.00174	1.3	0.085

implementation without entering specific details. As in the surface case, the cross section is represented in the form

$$\sigma = \sigma_I + \sigma_V \tag{9.15}$$

Table 9.4 Scattering parameters for different types of bottoms ($c_w = 1500\,\text{m/s}$, $\rho_w = 1\,\text{g/cm}^3$)

Bottom	γ_2	w_2	γ_3	w_3
Mud	3.25	0.0000164	3	0.00004269
Fine sand	3.25	0.000086	3	0.000354
Medium sand	3.25	0.0001406	3	0.000359
Coarse sand	3.25	0.00022	3	0.000362
Fine gravel	3	0.00018	3	0.000377
Gravel	3	0.00025	3	0.00152
Rock	3	0.00025	3	0.00006
Basalt	2.75	0.0001	3	0.0002

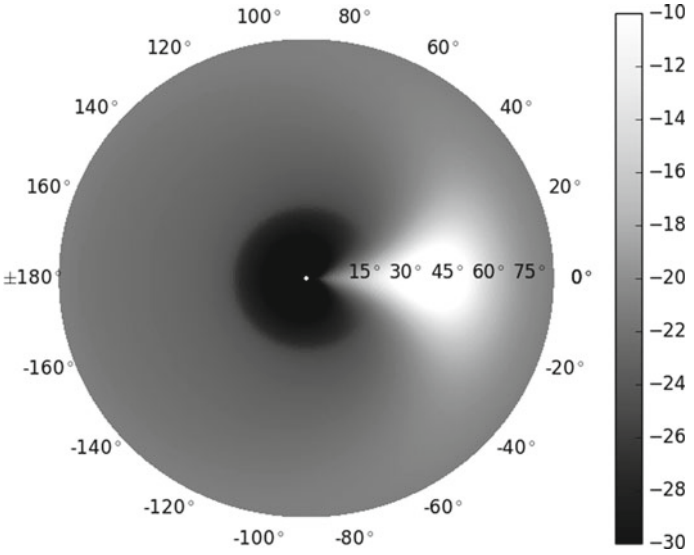


Fig. 9.3 Scattering strength for a sandy bottom (in dB)

where σ_I represents the contribution of boundary scattering between the water and the bottom and σ_V is the volume contribution inside the bottom. The calculation of σ_I is based on different approximations, namely, a Kirchhoff approximation for $\underline{\theta} \approx 90°$ and a composite roughness for the remaining angles. As in the surface case an interpolation formula allows to smooth the transition of one approximation to the other. The calculation of σ_V is based on a semi-empirical model, which requires the calculation of two factors that use the bottom reflection coefficient and the water/bottom impedances; the theoretical value is corrected adding an empirical parameter of volume scattering. Figure 9.3, generated by the Code Snippet 9.2, shows an example of the calculation of the scattering strength for a sandy bottom and a grazing angle of 45°; as expected scattering is maximal in the proximity of specular reflection ($\theta_s \approx \underline{\theta}$

and $\phi_s \approx 0°$), with losses being more pronounced for small scattering angles and a uniform scattering in the remaining directions.

Code Snippet 9.2 Scattering strength for a sandy bottom

```
from numpy import *
from scipy.special import *
from matplotlib.pyplot import *

rcParams.update({''text.usetex'': True})

def freflco(q=None,arho=None,vp=None,dp=None):

        a_p   = vp/( 1.0 + 1j*dp )
        apxap = a_p*a_p
        cosq  = cos( q )
        sinq  = sin( q )
        cqxcq = cosq*cosq
        sinqp = sqrt( 1.0 - apxap*cqxcq )
        zwp   = arho*a_p*sinq/sinqp
        Vww   = ( zwp - 1.0 )/( zwp + 1.0 )

        return Vww

def apl_bottom_bsigmav(qi = None, qs = None, phis = None,
        g3 = None, w3 = None, kw = None, arho = None,
        vp = None, dp = None):

        kappa = ( 1.0 + 1j*dp )/vp
        kappaxkappa = kappa*kappa

        cqs   = cos( qs   ); cqsxcqs = cqs*cqs
        cqi   = cos( qi   ); cqixcqi = cqi*cqi
        cphis = cos( phis )

        S = cqs*cqi*cphis

        pps = sqrt( kappaxkappa - cqsxcqs )
        ppi = sqrt( kappaxkappa - cqixcqi )

        Rs = freflco(qs,arho,vp,dp) + 1.0
        Ri = freflco(qi,arho,vp,dp) + 1.0
        RsxRs = abs( Rs )**2.0
        RixRi = abs( Ri )**2.0

        Repis = real( ppi + pps )
        Impis = imag( ppi + pps )
        A = 2.0*kw*arho*arho*Impis
        F = S - kappaxkappa - ppi*pps
        FxF = abs( F )**2.0
        dt = 0.5*sqrt( cqixcqi + cqsxcqs - 2.0*S )
        dk = kw*sqrt( 4.0*dt*dt + Repis*Repis )
        Wpp = w3/( dk**g3 )
        sigmav0 = pi/2.0*( kw**4.0 )*FxF*Wpp
        sigmav = sigmav0*RixRi*RsxRs/A

        return sigmav

def apl_bottom_bsigma(qi = None, qs = None, phis = None,
        g2 = None, w2 = None, kw = None, arho = None,
        vp = None, dp = None):
```

```
kappa = ( 1.0 + 1j*dp )/vp
kappaxkappa = kappa*kappa

alpha = g2/2.0 - 1.0
opa   = 1.0 + alpha
oma   = 1.0 - alpha
omta  = 1.0 - alpha*2.0
twoma = 2.0 - alpha
twoa  = 2.0*alpha
ooa   = 1.0/alpha
oota  = 1.0/twoa

G2ma = gamma( twoma )
Gopa = gamma( opa   )

cqs   = cos( qs   ); sqs = sin( qs )
cqsxcqs = cqs*cqs
cqi   = cos( qi   ); sqi = sin( qi )
cqixcqi = cqi*cqi
cphis = cos( phis )

Pi = sqrt( kappaxkappa - cqixcqi )
Ps = sqrt( kappaxkappa - cqsxcqs )

S = cqs*cqi*cphis

d  = 1.0/sqrt(2.0)*sqrt( 1.0 + sqi*sqs - S )
dz = 0.5*( sqi + sqs )
dt = 0.5*sqrt( cqixcqi - 2.0*S + cqsxcqs )

qis = math.asin( d )

dK = 2.0*kw*( dt + 0.0001 )

W2dK = w2/( dK**g2 )

Ris = freflco(qis,arho,vp,dp)
Ri  = freflco(qi ,arho,vp,dp) + 1.0
Rs  = freflco(qs ,arho,vp,dp) + 1.0

RxR   = abs( Ris )**2.0
RixRi = abs( Ri  )**2.0
RsxRs = abs( Rs  )**2.0

A = RxR/( 8.0*pi )
A1 = 8.0*alpha*alpha*gamma( oota + 0.5 )
A2 = gamma( 0.5 )*gamma( ooa )*gamma( oota )
a  = ( A1/A2 )**twoa
b = ( a**( 0.5 + 1.0/twoa )*gamma( ooa ) )/twoa
E = 2.0*pi*w2*G2ma*( 2.0**( -twoa ) )
F = alpha*oma*Gopa
ChxCh = E/F
q0 = ChxCh*( 2.0**( omta ) )*( kw**( 2.0*oma ) )
C  = ( dt**( 4.0*alpha ) + \
     a*q0*q0*( dz**4.0 ) )**( opa/twoa )
B = b*q0*( 1.0 + 100.0*dt**4.0 )/C
G = ( 1.0/arho - 1.0 )*\
( S - Pi*Ps/arho ) + 1.0 - kappaxkappa/arho
GxG = abs( G )**2.0

sigmakr = A*( d**4.0 )*B
sigmapr = 0.25*( kw**4.0 )*RixRi*RsxRs*GxG*W2dK

return sigmakr,sigmapr
```

```
freqkHz = 10.0
freq    = freqkHz*1000
cw      = 1500.0
kw      = 2.0*pi*freq/cw

# Sandy bottom:

arho    = 1.940
vp      = 1.113
dp      = 0.0115

g2      = 3.25
w2      = 0.000141
g3      = 3.0
w3      = 0.0004

eta = -2.0

thetai = 45; qi = thetai*pi/180

nthetas = 90
theta = linspace(1.0,90.0,nthetas)

nphis = 361
phis  = linspace(-180,180,nphis)

Sb = zeros((nphis,nthetas))
sigma = zeros((nphis,nthetas))

for j in range(nphis):
    phi = phis[j]*pi/180.0
    for i in range(nthetas):
        q = theta[i]*pi/180.0
        sigmav          = apl_bottom_bsigmav(qi,q,phi,
                          g3,w3,kw,arho,vp,dp)
        sigmakr,sigmapr = apl_bottom_bsigma( qi,q,phi,
                          g2,w2,kw,arho,vp,dp)
        sigmar = ( sigmakr**eta + \
                   sigmapr**eta )**( 1.0/eta )
        sigma[j,i]  = sigmav + sigmar
        Sb[j,i]  = 10*log10( sigmav + sigmar )

[R,Bearings] = meshgrid(theta,phis)

X = R*cos( Bearings*pi/180.0 )
Y = R*sin( Bearings*pi/180.0 )

figure(1)
pcolor(X,Y,Sb,cmap = 'Greys_r',vmin = -30,vmax = -10,shading = 'auto')
colorbar()
axis('off')

thetan = arange(15,90,15)
n = thetan.size
for i in range(n):
    s = str( thetan[i] ) + r'$^\circ$'
    text(thetan[i],0,s,color='black')
n = 18
for i in range(9):
    s = str( i*20 ) + r'$^\circ$'
    text(100*cos(i*2*pi/n),100*sin(i*2*pi/n),s,
    horizontalalignment = 'center',
    verticalalignment = 'center')

text(-100,0,r'$\pm180$^\circ$',
```

```
    horizontalalignment = 'center',
    verticalalignment = 'center')

    for i in range(9):
        s = str( -i*20 ) + r'$^\circ$'
        text(100*cos(i*2*pi/n),100*sin(-i*2*pi/n),s,
        horizontalalignment='center',
        verticalalignment='center')

    show()
```

9.6 Multibeam and Sidescan Sonars

Multibeam and sidescan sonars are systems of fundamental importance for bathymetric surveys. Often, their operation is illustrated by showing a vessel "illuminating" the seabed, thus revealing bottom topography as the vessel navigates along a given set of tracks. Independent tracks are later combined using numerical techniques in order to create a map of bottom depth in the surveyed area. The mentioned illustration can be somehow misleading because of the following reasons:

- In general, the source and receiving systems do not need to be located at the same position. There can be different configurations of sources and receivers, namely:
  - *Monostatic*: the source and receiving systems are located at the same position;
  - *Bistatic*: the source system is located at one position and the receiving system at another;
  - *Multistatic*: the source and receiving systems are located at different positions.
- Such illustration can imply that acoustic signals are being *reflected* back to the receiving system, while in most cases the returning mechanism is due to bottom scattering. In fact, a large amount of energy is lost through reflections in the specular directions and the energy that echoes back to the receiver system is rather weak, and requires amplification.
- The illuminating beam is shown rather straight on the edges, while sound refraction effects can be expected to become more relevant as the beam widens.
- In order to achieve the desired spatial resolution emitted signals need to be of high frequencies (often tenths or hundreds of kHz); at such frequencies signals quickly attenuate as they propagate to the bottom, and this attenuation sets important constraints to the vertical sensitivity of the sonar system.
- Multibeam and sidescan sonars are not always hull-mounted. Some devices (often called "fish") can be towed behind the vessel, and their proximity to the bottom can be regulated for a better sweeping of the bathymetry.

- A multibeam sonar uses a set of acoustic sources, and sweeps a given area uniformly. A sidescan sonar uses a pair of sources, placed symmetrically on the sides of the system, both producing a cone-shaped slanted beam, leaving the area just below the sonar system in the shadow.

The discussion of multibeam and sidescan sonar systems is rather technical and there is no lack of literature on the subject (see, for instance, [68, 69]). Yet, very often, the discussion addresses few aspects of signal transmission and reception as a problem of wave propagation; for such context a particular sonar configuration could be idealized as a problem of ray tracing, combined with scattering calculations for a particular bottom type and for a given set of incident and scattering angles [70]; such preliminary assessment is still under development and can be expected to receive more attention in the future.

9.7 Boundary Reverberation

Boundary reverberation is given by [20]

$$\underline{\mathrm{RL}} = 10 \log V \tag{9.16}$$

where V represents the reverberation volume, which corresponds to

$$V = \frac{1}{2} c\tau\Phi r \tag{9.17}$$

where r represents the distance from the array to the target, c is the mean speed of sound, τ is the duration of the transmitted pulse and Φ represents the equivalent angle, which depends on the type of array and it is calculated integrating the beam pattern of the transmitting array, multiplied by the beam pattern of the receiving array. In the case of a point array $\Phi = 2\pi$.

9.8 Example

Reverberation manifests itself in the received signal as an intensity background that decays progressively. This decay is illustrated in Fig. 9.4, which is generated by the Code Snippet newinlinkFigclst:examplerev9.3; the figure shows that the reverberation level as a function of time in a waveguide with constant sound speed; the calculations consider a pulse with a duration of one second. To simplify the code the bottom cross section is idealized instead of being calculated directly.

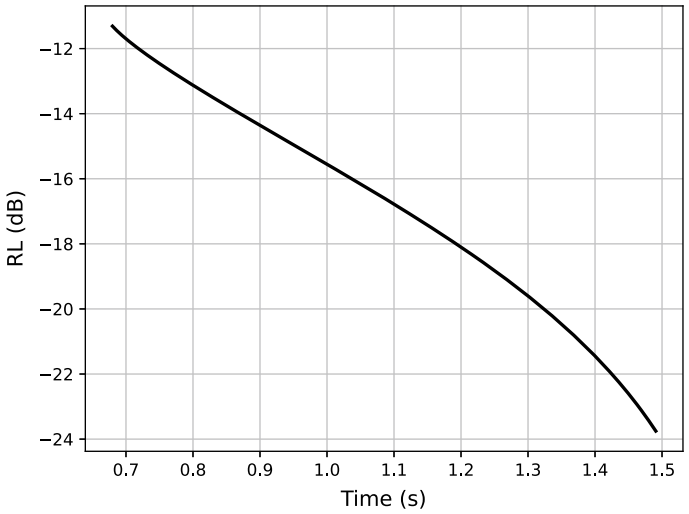


Fig. 9.4 Reverberation level in an isovelocity waveguide as a function of time

Code Snippet 9.3 Reverberation level as a function of time

```
from numpy import *
from matplotlib.pyplot import *

# Parameters
SL   = 0
phi  = 2*pi
tau  = 1 # Pulse duration
freq    = 3000
Dmax    = 1000
zs      = 500.0
cw      = 1500.0

nr = 501
rmin =  100.0
rmax = 1000.0
r = linspace(rmin,rmax,nr)

s1 = 0.016
s2 = 0.002
sigma = linspace(s1,s2,nr) # Idealization

trev   = ones(nr)
RL     = ones(nr)

for i in range(nr):
    ri = sqrt(r[i]*r[i] + (Dmax-zs)**2.0)
    TL = 10*log10( abs( ri ) )
    A = 0.5*cw*tau*pi*ri
    RLb = 10*log10(A)
    RL[i] = SL - 2*TL + 10*log10( sigma[i] ) + RLb
```

```
    trev[i] = 2*ri/cw

figure(1)
plot(trev,RL,'k',linewidth=2)
xlabel('Time (s)',fontsize=18)
ylabel('RL (dB)'  ,fontsize=18)
grid(True)

show()
```

Chapter 10
Inverse Problems

10.1 Introduction

An *inverse problem*, within the context of Underwater Acoustics, is an application that often involves a propagation model, but can be expanded well beyond the calculation of transmission loss; in an inverse problem one takes advantage of the processing of acoustic data to retrieve information regarding the ocean environment. Since the intrinsic nature of inverse problems is highly multi-parametric (and seldom admits unique solutions) the different applications of inverse problems are adapted to subsets of different parameters, which in the general case address the following problems:

- *Source localization* with either a horizontal or a vertical array, excluding *triangulation* methods (which require a simultaneous and collaborative reception of acoustic signals at different locations), but allows to include (in a rather broad sense) the problem of *Direction Of Arrival* (to be addressed in Sect. 10.2).
- *Geoacoustics*: to characterize the sea bottom by acoustic means, to be addressed in Sect. 10.3.
- *Tomography*: to map by acoustic means the temperature field of a volume of the watercolumn, to be addressed in Sect. 10.4.

In essence all these problems are addressed in a similar manner, namely, through the definition of a *cost function* (also known as an *ambiguity function*), which "measures" the similarity between the measured and predicted fields for a provable set of parameters; the parameter set that optimizes the cost function corresponds to the solution of the inverse problem. The sections that follow will be dedicated to the discussion of each inverse problem.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023

O. C. Rodríguez, *Fundamentals of Underwater Acoustics*,
https://doi.org/10.1007/978-3-031-31319-6_10

10.2 Source Localization

Source localization is the inverse problem of determining the three-dimensional position of an acoustic source, provided that bottom properties and sound speed are known. Such localization is feasible from a numerical point of view [71]; however, even with today's available computational power, there is no guarantee that the problem can be solved as fast as desired. In order to simplify the discussion, we will consider two variants of the problem, namely,

- The problem of determining the *Direction Of Arrival* (abbreviated as *DOA*) of the signal transmitted by a source, relative to a linear array.
- The problem of determining source's range and depth relative to a vertical array.

Each of these problems is addressed in the following sections.

10.2.1 Bearing Estimation

Let us consider a hydrophone array with a vector spacing $\mathbf{d} = d\mathbf{e}_a$ between hydrophones, where $\mathbf{e}_a$ is a unit vector that defines the array alignment; for instance, $\mathbf{d} = d\mathbf{e}_x$ corresponds to a linear array aligned along the X-axis. On this array arrives a plane wave, with a wavenumber given by

$$\mathbf{k}_s = -\frac{\omega}{c}\begin{bmatrix} \cos(\phi_s)\cos(\underline{\theta}_s) \\ \sin(\phi_s)\cos(\underline{\theta}_s) \\ \sin(\underline{\theta}_s) \end{bmatrix}$$

where $\underline{\theta}$ represents the *elevation* and ϕ represents the *azimuth* (see Fig. 10.1). At the array the phase delay between each hydrophone is given by

$$y_l = \exp\left(-i\mathbf{k}_s^T\mathbf{r}_l\right)$$

where $\mathbf{r}_l = (l-1)\mathbf{d}$ and $l = 1, 2\ldots$, which can itself be represented with a vector of the arriving wave $\mathbf{y}$. From the point of view of the array, one can consider the sweeping vector $\mathbf{e}$, with components given by

$$e_l = \exp\left(i\mathbf{k}^T\mathbf{r}_l\right)$$

where

$$\mathbf{k} = \frac{\omega}{c}\begin{bmatrix} \cos(\phi)\cos(\underline{\theta}) \\ \sin(\phi)\cos(\underline{\theta}) \\ \sin(\underline{\theta}) \end{bmatrix}$$

The *beamforming* can now be defined as

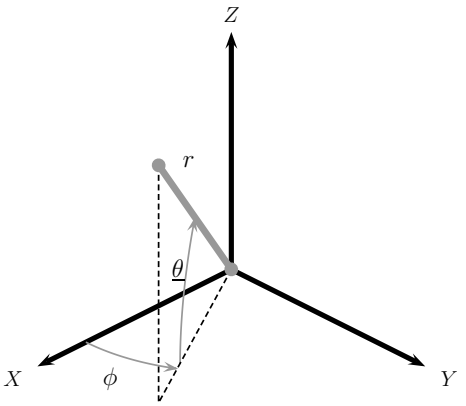


Fig. 10.1 Angular coordinates of elevation and azimuth

$$B(\phi, \underline{\theta}) = \left|\mathbf{e}^{*}\mathbf{y}\right| \tag{10.1}$$

and its maximum indicates the direction from which the source transmitted the arriving signal.

Within the context of a linear array one should not expect that the localization of the source will proceed without ambiguities. For instance, for the case of a vertical array ($\mathbf{d} = d\mathbf{e}_z$) the beamforming "misses" all information related to the horizontal components of $\mathbf{k}_s$ and remains "blind" relative to the azimuth. This ambiguity problem is illustrated in Fig. 10.2, generated by the Python Code Snippet 10.1, which shows the beamforming of a vertical array for a wave arriving with a direction given by $\phi_s = 20°$ and $\theta_s = 40°$. As expected, the beamforming correctly infers the elevation, but is unable to determine the source's azimuth. Different array alignments can mitigate the ambiguity, but it cannot be eliminated; as discussed in [36] one needs to resort to arrays with non-linear configurations to improve the angular resolution of the array, or to extend the beamforming algorithm to process vector sensor data acquired in a linear array. Real scenarios are seldom symmetric in azimuth relative to a vertical array, but the cases of practical interest make extremely difficult to consider the arrival of a plane wave on an array because the speed of sound is not constant over depth; on another side DOA determination can be considered using short or horizontal arrays.

Code Snippet 10.1 Beamforming

```
from numpy import *
from matplotlib.pyplot import *

rcParams.update({''text.usetex'': True})

ntheta = 181
theta = linspace(-90,90,ntheta)
```

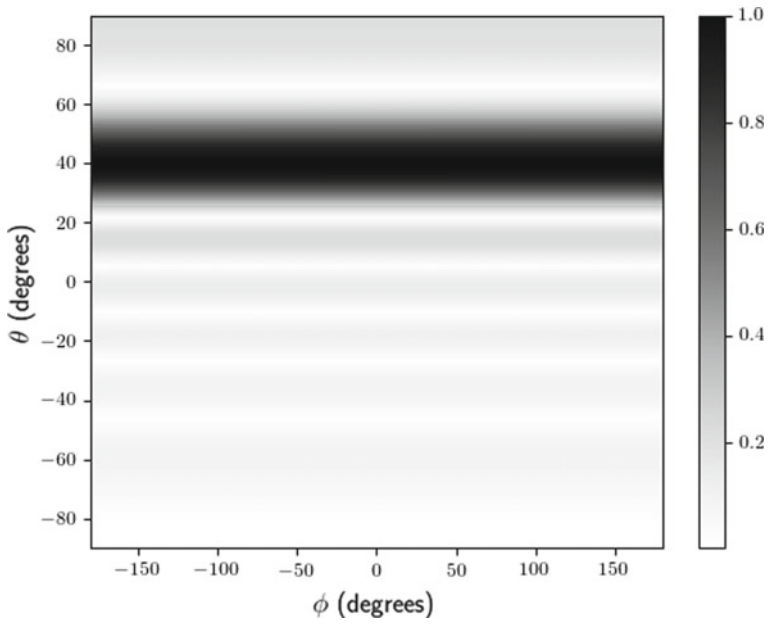


Fig. 10.2 Beamforming for a vertical array with 11 hydrophones, for a wave arriving from the direction given by $\phi_s = 20°$ and $\theta_s = 40°$; $f = 500$ Hz, $c = 1500$ m/s, $d = 1$ m

```
thetar = theta*pi/180
nphi = 361
phi = linspace(-180,180,nphi)
phir = phi*pi/180

thetas  = 40
thetasr = thetas*pi/180;
phis    = 20
phisr   = phis*pi/180;

nsensors = 11
c    = 1500
freq =  500; w0 = 2*pi*freq
d = array([0, 0, 1]) # Vertical array

yp = zeros(nsensors) + 1j
ep = zeros(nsensors) + 1j
B  = zeros((ntheta,nphi))

kx = cos(phisr)*cos(thetasr)
ky = sin(phisr)*cos(thetasr)
kz =            sin(thetasr)
ks = -w0/c*array([kx,ky,kz])

for l in range(nsensors):
    rl = l*d
```

```
        yp[l] = exp( -1j*dot(ks,rl) )

    yp = yp/linalg.norm( yp )

    for j in range(nphi):

        phij = phir[j]

        for i in range(ntheta):

            thetai = thetar[i]
            kx = cos(phij)*cos(thetai)
            ky = sin(phij)*cos(thetai)
            kz =           sin(thetai)
            k  = w0/c*array([kx,ky,kz])

            for l in range(nsensors):

                rl = l*d
                ep[l] = exp( 1j*dot(k,rl) )

            ep = ep/linalg.norm( ep )

            B[i,j] = abs( dot(conj(ep),yp) )

    figure(1)
    pcolor(phi,theta,B,cmap='Greys',shading='auto')
    colorbar()
    xlim(-180,180)
    ylim(-90,90)
    xlabel(r'$\phi$ (degrees)'  ,fontsize=18)
    ylabel(r'$\theta$ (degrees)',fontsize=18)

    show()
```

10.2.2 Range and Depth Estimation

The beamforming discussed in the previous section is based on a plane wave approximation arriving at the array in order to determine the source azimuth and elevation relative to the array, but it is completely independent of the relative distance between the array and the source. In the case of a vertical array, we can resort to a propagation model for the calculation of field predictions for provable values of source range and depth, and compare each prediction with a field observation; the prediction that is found to be more "similar" to the observation will correspond to the unknown values of source depth and range. In a first approach to the comparison, one might consider the absolute value of the integral difference between the observation and the prediction, but this approach would be rather inefficient. In fact, as suggested by Eq. (10.1), it would be more efficient to measure such similarity through

Matched-Field Processing (abbreviated as *MFP*), that is, to resort to an inner product between the observation **d** and the model prediction **e**, after normalization:

$$B(r, z) = \mathbf{e}^* \mathbf{d}$$

The concept of MFP, being initially suggested in the context of source localization, became a synonym of localization itself, but its general principle can be applied to other inverse problems. Since observations are contaminated with noise one needs to calculate first the covariance matrix of the data:

$$\mathbf{R} = \frac{1}{L} \sum_{l=1}^{L} \mathbf{d}\mathbf{d}^* \tag{10.2}$$

where L represents the number of *snapshots*; this matrix is then used in the cost function. Let us consider, as a reference example, the *Bartlett estimator* defined as

$$B(r, z) = \mathbf{e}^* \mathbf{R} \mathbf{e} \tag{10.3}$$

Once the observation and the prediction are both normalized the value of the Bartlett estimator remains between 0 and 1. For broadband signals one can consider an incoherent estimator given by

$$B = \frac{1}{N} \sum_{j=1}^{N} B_i \tag{10.4}$$

where N represents the number of frequencies and each B_i is given by Eq. (10.5).

Source localization, using Bartlett's estimator, is illustrated in Fig. 10.3 in the case of an isovelocity waveguide; the figure is produced by the Python Code Snippet 10.2. To simplify the code the calculations are performed in a noiseless case, considering a vertical array with 51 hydrophones between depths of 25–75 m and a frequency 50 Hz; as expected, the Bartlett estimator exhibits a clear maximum at the "real" position of the source, which was taken at a range of 1 km and a depth of 50 m; it is worth noticing that despite the idealization of a noiseless environment the cost function exhibits numerous "concurrent" maxima.

Code Snippet 10.2 Source localization with the Bartlett estimator

```
from numpy import *
from scipy.interpolate import interp1d
from matplotlib.pyplot import *

def pmodes(z = None, Z = None, k = None,
           zs = None, zh = None, rh = None, M = None):

    p = []
```

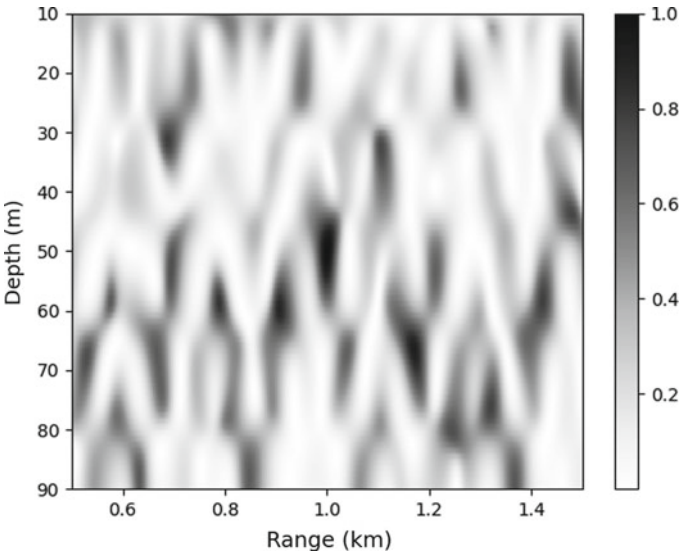


Fig. 10.3 Bartlett estimator for a source at the position $r_s = 1$ km and $z_s = 50$ m

```
    nr = rh.size
    nz = zh.size

    p  = zeros((nz,nr)) + 1j*zeros((nz,nr))
    pm = zeros((nz,nr)) + 1j*zeros((nz,nr))

    e0 = 1j*exp( -1j*pi/4.0 )/sqrt( 8*pi )

    dz = abs( z[1] -z[0] )

    for i in range(M):
        Zm = Z[:,i]
        interpolator = interp1d(z, Zm)
        Zms = interpolator( zs )
        Zmh = interpolator( zh )
        Zsh = Zms*Zmh
        Rm  = exp( 1j*k[i]*rh )/sqrt( k[i]*rh )
        [RM,ZM] = meshgrid(Rm,Zsh)
        pm  = RM*ZM
        p = p + pm

    p = e0*p
    p = squeeze( p )

    return p

zs =  50.0
rs = array([1000.0])

nr   = 201
```

```
rmin =  500.0
rmax = 1500.0
rb = linspace(rmin,rmax,nr); rkm = rb/1000.0
nz   = 101
D    = 100.0
z    = linspace(0.0,D,nz)
nzb = 81
zb = linspace(10,90,nzb)
c = 1500.0
freq = 50.0; w = 2*pi*freq
k = w/c; kxk = k*k
M = int( w*D/(pi*c) - 0.5 )
nh = 51
zh = linspace(25,75,nh)
km = ( arange(0,M) + 0.5 )*pi/D
kmsq = km*km
kh = sqrt( kxk - kmsq )

Z = zeros((nz,M))

for i in range(M):
    Zs = sin( km[i]*zs )
    Z[:,i]  = sin( km[i]*z )

Z = 2.0/D*Z

d = pmodes(z,Z,kh,zs,zh,rs,M)
n = linalg.norm( d )
dn = conj( d/n )

B = zeros((nzb,nr))

for j in range(nzb):
    zj = zb[j]
    e = pmodes(z,Z,kh,zj,zh,rb,M)
    for i in range(nr):
        ei = e[:,i]
        n = linalg.norm(ei)
        en = ei/n
        prod = inner( en, dn )
        B[j,i] = abs( prod*conj( prod ) )

figure(1)
imshow(B, cmap='Greys',
extent=[rkm[0],rkm[-1],zb[-1],zb[0]],
aspect='auto',origin='lower')
colorbar()
xlabel('Range (km)',fontsize=18)
ylabel('Depth (m)' ,fontsize=18)

show()
```

The Bartlett estimator is one of many possible choices; for instance, range and depth estimation can be based also on the *Capon estimator* (also known as *minimum variance* or *maximum likelihood method*), given by [8]:

$$C(r,z) = \frac{1}{\mathbf{e}^{*}\mathbf{R}^{-1}\mathbf{e}} \tag{10.5}$$

Sometimes, the calculation of the inverse matrix is numerically unstable; *diagonal overloading* (described in Appendix B) can be used to fix the instability. An interesting (model-based) approach to range and depth estimation is based on *modal decomposition* [72], which can be described as follows: without loss of generality let us write the pressure field as a function of depth z as

$$p(z) = \sum_{m} Z_m(z) a_m(r_s, z_s)$$

where

$$a_m(r_s, z_s) = \left(\frac{2\pi}{\underline{k}_m r_s}\right)^{1/2} Z_m(z_s) \exp\left(i\underline{k}_m r_s\right)$$

r_s is the source range, z_s is the source depth, and $Z_m(z)$ is the mth normal mode. At a particular depth z_j the pressure field can be written as

$$p(z_j) = \mathbf{E}_j \mathbf{a}$$

where

$$\mathbf{a} = \begin{bmatrix} a_1(r_s, z_s) \\ a_2(r_s, z_s) \\ \cdots \\ a_m(r_s, z_s) \end{bmatrix} \quad \text{and} \quad \mathbf{E}_j = \begin{bmatrix} Z_1(z_j) \\ Z_2(z_j) \\ \cdots \\ Z_m(z_j) \end{bmatrix}^T$$

For a given set of depths $z_1, z_2, \ldots z_N$ we can write an observation as

$$\mathbf{p} = \mathbf{E}\mathbf{a}$$

where

$$\mathbf{p} = \begin{bmatrix} p(z_1) \\ p(z_2) \\ \cdots \\ p(z_N) \end{bmatrix} \quad \text{and} \quad \mathbf{E} = \begin{bmatrix} Z_1(z_1) & Z_2(z_1) & \ldots & Z_m(z_1) \\ Z_1(z_2) & Z_2(z_2) & \ldots & Z_m(z_2) \\ \vdots & \vdots & \ddots & \vdots \\ Z_1(z_N) & Z_2(z_N) & \ldots & Z_m(z_N) \end{bmatrix}$$

Providing that $N > m$ we can obtain the modal amplitudes of the corresponding observation through the *least-squares solution* (see Appendix B), given by

$$\mathbf{a} = \left(\mathbf{E}^T\mathbf{E}\right)^{-1}\mathbf{E}^T\mathbf{p}$$

Therefore, the "modal" estimator can be written as

$$M(r,z) = \left|\mathbf{a}(r,z)^{*}\mathbf{a}(r_s,z_s)\right|^2$$

where the modal amplitudes $\mathbf{a}(r,z)$ can be calculated from field predictions $\mathbf{p}(z)$ for a given set of candidate ranges r and depths z. Normalizing both the observed and predicted modal amplitudes will produce an estimator with values between 0 and 1. Python coding for range and depth estimation based on modal decomposition is left to the reader as an exercise.

10.3 Geoacoustics

Geoacoustics is the inverse problem in which acoustic data is processed in order to determine the bottom properties, as long as the source position and the sound speed are well known. This is obviously a multi-parametric problem given the large set of parameters that define the bottom, which is often organized into a layered structure. Exhaustive calculation of the cost function (as opposed to the previous case) is often rather inefficient or simply impossible. The alternative consists in resorting to optimization strategies which sometimes mimic natural processes, such as genetic algorithms [46]; such optimization relies on statistical distributions of the parameters, which can be used to determine the accuracy of the results. Figure 10.4 illustrates the

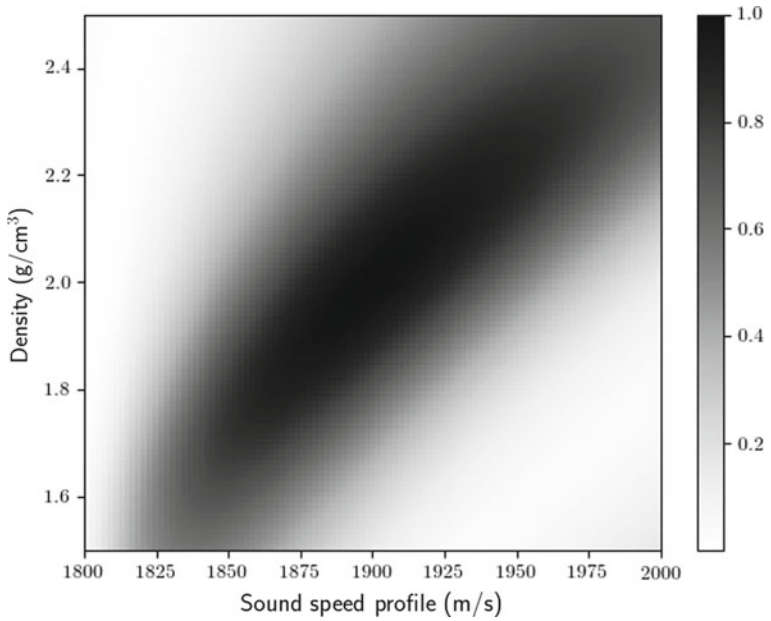


Fig. 10.4 Bartlett estimator for bottom properties (vertical array with 51 hydrophones)

result of a geoacoustic inversion with the Bartlett estimator, in an idealized situation which considers a Pekeris waveguide; thus, calculations require only to estimate the speed of sound in the bottom and the corresponding density. Unlike the results of the previous section the figure reveals a rather broad maxima of the cost function, which indicates a clear correlation between bottom sound speed and bottom density.

Code Snippet 10.3 Bottom characterization with the Bartlett estimator

```
from numpy import *
from scipy.interpolate import interp1d
from matplotlib.pyplot import *

rcParams.update({"text.usetex": True})

def kpekeris(c1=None,rho1=None,
             c2=None,rho2=None,
             D=None,freq=None):

    kp = []

    w = 2*pi*freq
    k1 = w/c1
    k2 = w/c2

    nk = 1001
    k = linspace(k1,k2,nk)
    kxk = k*k

    r1 = sqrt( k1*k1 - kxk )
    r2 = sqrt( kxk - k2*k2 )
    s1 = sin( r1*D )
    c1 = cos( r1*D )

    f = s1*r2/rho2 + c1*r1/rho1

    for i in range(nk-1):

        x1 = k[ i ]
        x2 = k[i+1]
        y1 = f[ i ]
        y2 = f[i+1]
        p = y1*y2
        x3  = (x1*y2-x2*y1)/(y2-y1)

        if p < 0:

           kp = append(kp,x3)

    return kp

def pmodes(z=None,Z=None,k=None,
           zs=None,zh=None,rh=None,M=None):

    p = []
    nr = rh.size
```

```
    nz = zh.size

    p  = zeros((nz,nr)) + 1j*zeros((nz,nr))
    pm = zeros((nz,nr)) + 1j*zeros((nz,nr))

    e0 = 1j*exp( -1j*pi/4.0 )/sqrt( 8*pi )

    dz = abs( z[1] -z[0] )

    for i in range(M):
        Zm = Z[:,i]
        interpolator = interp1d(z, Zm)
        Zms = interpolator( zs )
        Zmh = interpolator( zh )
        Zsh = Zms*Zmh
        Rm  = exp( 1j*k[i]*rh )/sqrt( k[i]*rh )
        [RM,ZM] = meshgrid(Rm,Zsh)
        pm  = RM*ZM
        p = p + pm

    p = e0*p
    p = squeeze( p )

    return p

freq   = 50.0; w = 2*pi*freq
# Expected values:
cb     = 1900.0; rhob = 2.0
# Waveguide:
D      =  100.0
c1     = 1500.0; rho1 = 1.0
# Bartlett parameters:
nc     = 101
cmin = 1800.0
cmax = 2000.0
c2     = linspace(cmin,cmax,nc)
nrho = 101
rhomin = 1.5
rhomax = 2.5
rho2   = linspace(rhomin,rhomax,nrho)
# Waveguide:
zs     = 50.0
rh     = array([5000.0])
B      = zeros((nrho,nc))
nzh    = 51
zh     = linspace(25,75,nzh)
z      = linspace(0,D,101); dz = abs( z[1]-z[0] )

# Observation
# Pekeris: wavenumbers
kh     = kpekeris(c1,rho1,cb,rhob,D,freq)
M      = kh.size
Z      = zeros((101,M))
kz     = sqrt( (w/c1)**2 - kh*kh )
# Modes
for j in range(M):
    Zj = sin( kz[j]*z )
    N = sqrt( dz*sum( Zj*Zj ) )
```

```
    Z[:,j] = Zj/N
# Modes field
d = pmodes(z,Z,kh,zs,zh,rh,M)
n = linalg.norm(d)
dn = d/n

# Replica calculation and field matching:
for j in range(nc):
    ci = c2[j]
    for j2 in range(nrho):
        rhoi = rho2[j2]
        kh = kpekeris(c1,rho1,ci,rhoi,D,freq)
        M = kh.size
        Z = zeros((101,M))
        kz = sqrt( (w/c1)**2 - kh*kh )
        for jj in range(M):
            Zj = sin( kz[jj]*z )
            N = sqrt( dz*sum( Zj*Zj ) )
            Z[:,jj] = Zj/N
        eij = pmodes(z,Z,kh,zs,zh,rh,M)
        n = linalg.norm( eij )
        en = eij/n
        prod = inner( en , conj(dn) )
        B[j2,j] = abs( prod*conj(prod) )

figure(1)
imshow(B, cmap='Greys',
extent=[c2[0],c2[-1],rho2[0],rho2[-1]],
aspect='auto',origin='lower')
colorbar()
xlabel('Sound speed profile (m/s)',fontsize=18)
ylabel(r'Density (g/cm$^3$)',fontsize=18)
show()
```

10.4 Tomography

Tomography is the inverse problem in which we intend to retrieve information regarding the sound speed *field* from the processing of acoustic data, given a known source position and knowing the bottom properties. As initially proposed, tomography explores the variations of arrival times, $\Delta\tau$, to determine the perturbations of sound speed relative to a reference profile [73]:

$$\delta c(z) = c(z) - c_0(z)$$

This perspective of the tomographic problem, in which the main characteristic of the signal to be considered corresponds travel-time data, is known as *travel-time tomography*. Alternatively, one can adapt the methods of MFP to determine directly

the sound speed, in what is known as *matched-field tomography*. The following sections will discuss these two approaches to the tomographic problem.

10.4.1 Matched-Field Tomography

Let us consider an orthogonal basis of functions, which allows to parametrize the sound speed profile:

$$c(z) = c_0(z) + \sum_m \alpha_m E_m(z) \tag{10.6}$$

In matched-field tomography, the inverse problem is then solved when the cost function is optimized for a particular set of parameters

$$\boldsymbol{\alpha} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \cdots \\ \alpha_m \end{bmatrix} \tag{10.7}$$

Within this context it is important to take into account that

- The components of the vector $\boldsymbol{\alpha}$ are not all of equal importance; eventually, some of the components might be *non-observable*, meaning that their impact on the variations of the acoustic field is very small or can be neglected.
- Matched-field tomography is only reliable when the array allows to perform a proper sampling of the acoustic field in depth.

A natural choice for the set of orthogonal functions corresponds to the *empirical orthogonal functions* (abbreviated as *EOFs*), which can be calculated as follows:

- Let us consider a set of discrete dataset with samples of sound speed $\mathbf{c}_1$, $\mathbf{c}_2$, ..., $\mathbf{c}_N$, where $N > M$, and M is the number of points in each discretized profile.
- We proceed with the calculation of the mean profile:

$$\mathbf{c}_0 = \frac{1}{N} \sum_{k=1}^{N} \mathbf{c}_k$$

- The mean is then used to calculate a matrix of sound speed perturbations:

$$\delta\mathbf{C} = [\delta\mathbf{c}_1 \ , \ \delta\mathbf{c}_2 \ , \ \ldots \ , \ \delta\mathbf{c}_n]$$

  where $\delta\mathbf{c}_k = \mathbf{c}_k - \mathbf{c}_0$.
- The next step is to calculate the *singular value decomposition* (abbreviated as *SVD*) of the matrix $\delta\mathbf{C}$:

$$\delta\mathbf{C} = \mathbf{U} \times \mathbf{S} \times \mathbf{V}^T$$

where **S** is a diagonal matrix, which contains the eigenvalues of the SVD, and

- The EOFs are given by the columns of **U**.
- Each eigenvalue of **S** is representative of the "importance" of the corresponding EOF.

Alternatively, it is possible to consider other choices of orthogonal functions, such as the hydrodynamic modes [73] or Chebyshev polynomials [74].

10.4.2 Travel-Time Tomography

The starting point for the discussion of travel-time tomography is given by the expression

$$\Delta\tau = \int_{\zeta} \frac{d\zeta}{c(z)} - \int_{\zeta_0} \frac{d\zeta}{c_0(z)} \tag{10.8}$$

where ζ and ζ_0 represent eigenrays that correspond to the sound speed $c(z)$ and to the reference sound speed $c_0(z)$, respectively. In the case of small perturbations

$$\delta c(\mathbf{r}) = c(\mathbf{r}) - c_0(\mathbf{r}) \ll c_0(z) \tag{10.9}$$

it can be considered that $\zeta \approx \zeta_0$, which allows to rewrite Eq. (10.8) in the form

$$\Delta\tau_j = \tau_j - \tau_j^0 = \int_{\zeta_j} \frac{d\zeta}{c(\mathbf{r})} - \int_{\zeta_j^0} \frac{d\zeta}{c_0(\mathbf{r})} \approx -\int_{\zeta_j^0} \frac{\delta c(\mathbf{r})}{c_0^2(\mathbf{r})} d\zeta \tag{10.10}$$

In this expression the integral is calculated along a stable eigenray. "Collecting" a set of N perturbations in the arrival times and representing the acoustic waveguide as a system composed of L layers, it is possible to obtain the following system of linear equations [73]:

$$\mathbf{y} = \mathbf{E}\mathbf{x} + \mathbf{n} \tag{10.11}$$

where $\mathbf{y} = [\ \Delta\tau_1\ \Delta\tau_2\ \ldots\ \Delta\tau_N\ ]^T$, $\mathbf{x} = [\ \delta c_1\ \delta c_2\ \ldots\ \delta c_L\ ]^T$, in which each δc_l corresponds to the mean value of $\delta c(z)$ in the layer of index l. The vector $\mathbf{y}$ is known as the *vector of temporal delays* or simply the *delay vector*. In Eq. (10.11) $\mathbf{n}$ represents the noise contribution to the set of observations $\mathbf{y}$. The matrix $\mathbf{E}$ is known as the *observation matrix*; there is no general rule to choose a number of layers, but in the inversion one needs to take into account that the observation matrix can be rank deficient [48]. This issue can be mitigated resorting to the *regularization* of the system of equations, which consists in taking advantage of the expansion given by Eq. (10.6) to relate the delay vector to the amplitudes of the basis functions [73]. The solution of Eq. (10.11) is described in detail in Appendix B.

Appendix A
Sea Surface Roughness

A.1 Introduction

The shape of the sea surface can be wind-driven and/or modulated by swells; in the first case, one needs to take into account wavenumber and angular spectra; a brief characterization of spectra and swells is provided in the following sections.

A.2 Spectra of Wind-Driven Sea Surface Waves

A characterization of a wind-driven sea surface is given by the Pierson–Moskowitz spectrum [75, 76]:

$$S(f) = \alpha \frac{g^2}{(2\pi)^4 f^5} \exp\left[-\frac{5}{4}\left(\frac{f_m}{f}\right)^4\right] \tag{A.1}$$

where

$$f_m = 0.13 \frac{g}{v} \tag{A.2}$$

is the peak's frequency, v represents the wind velocity, $g = 9.8$ m/s^2 and $\alpha = 8.1 \times 10^{-3}$ is Phillip's constant; the quantity

$$T_p = \frac{1}{f_m} = \frac{v}{0.13\, g} \tag{A.3}$$

represents the peak's period. The Pierson–Moskowitz spectrum is intended to represent a fully developed wind generated sea and it is valid in the interval $10\,\text{m/s} < v < 20\,\text{m/s}$. Another alternative is given by the JONSWAP spectral model [77]:

© The Editor(s) (if applicable) and The Author(s), under exclusive license to Springer Nature Switzerland AG 2023

O. C. Rodríguez, *Fundamentals of Underwater Acoustics*,
https://doi.org/10.1007/978-3-031-31319-6

$$S(\omega) = \alpha' g^2 \omega^{-5} \exp\left[-\frac{5}{4}\left(\frac{\omega}{\omega_p}\right)^{-4}\right]\gamma^{\delta} \tag{A.4}$$

where δ is a peak enhancement factor:

$$\delta = \exp\left[-\frac{\left(\omega - \omega_p\right)^2}{2\sigma_0^2 \omega_p^2}\right] \tag{A.5}$$

The parameters γ and σ_0 are given as $\gamma = 3.3$, $\sigma_0 = 0.07$ for $\omega \leq \omega_p$ and $\sigma_0 = 0.09$ for $\omega > \omega_p$, while α' is function of fetch X and wind speed v:

$$\alpha' = 0.076\left(\frac{gX}{v}\right)^{-0.22} \tag{A.6}$$

and peak frequency ω_p is given by

$$\omega_p = 7\pi\left(\frac{g}{v}\right)\left(\frac{gX}{v^2}\right)^{-0.33} \tag{A.7}$$

A.2.1 Wave Height and Speed

Surface wave height can be related to wind speed through the well-known Beaufort scale. For deepwater propagation the phase velocity of the surface wave is calculated as

$$v_p = \frac{g}{\omega_p} \tag{A.8}$$

while the group velocity corresponds to

$$v_g = \frac{1}{2}v_p \tag{A.9}$$

A.2.2 Frequency to Wavenumber Conversion

The spectra previously described are defined in the frequency domain. For acoustic propagation, it is more convenient to generate realizations of the sea surface in space rather than in time, which requires the spectra to be defined in the wavenumber domain. The conversion from frequency to wavenumber can be accomplished by taking into account that

$$E(k)dk = S(f)df \tag{A.10}$$

where $S(f)$ represents the spectrum in the frequency domain and $E(k)$ represents the spectrum in the wavenumber domain. For surface waves over deep oceans frequency and wavenumber can be related as

$$\omega = \sqrt{gk} \tag{A.11}$$

which allows to obtain the relationship

$$\frac{df}{dk} = \frac{1}{4\pi}\sqrt{\frac{g}{k}} \tag{A.12}$$

Thus, the spectrum in the wavenumber domain can be calculated from the spectrum in the frequency domain as

$$E(k) = \frac{1}{4\pi}\sqrt{\frac{g}{k}}\, S(f) \tag{A.13}$$

Generally speaking, the peak spectrum shifts towards lower wavenumbers (i.e. longer wavelengths) as wind speed increases. Therefore, high values of wind speed induce the propagation of low-frequency surface waves, associated with larger amplitudes and more energy transfer to the ocean.

A.2.3 Angular Spreading

The directivity of wind speed can be incorporated into the modelling by introducing a spreading function along different angles; the full spectrum can then be written as

$$F(k, \theta) = E(k) D(f, \theta)\ , \tag{A.14}$$

where $D(f, \theta)$ represents the spreading function. Since the total energy in the directional spectrum should be the same as the total energy in the one-dimensional spectrum a fundamental condition for the choice of the function $D(f, \theta)$ is that

$$\int_{-\infty}^{\infty}\int_{-\pi}^{\pi} E(k) D(k, \theta)\, d\theta\, dk = \int_{-\infty}^{\infty} E(k)\, dk \tag{A.15}$$

Some of the most common spreading functions are presented in the following sections.

Cosine-Squared Spreading Function

The cosine-squared sea surface spreading (which is independent of frequency) is given by the expression [78, 79]

$$D(\theta) = \begin{cases} \frac{2}{\pi}\cos^2(\theta - \theta_0) & -\frac{\pi}{2} + \theta_0 < \theta < \frac{\pi}{2} + \theta_0 \,, \\ 0 & \text{otherwise} \,, \end{cases} \tag{A.16}$$

where θ_0 represents the wind's direction. The cosine-squared spreading function is strongly non-isotropic and different from zero only in the region where $|\theta - \theta_0| < \pi/2$.

Mitsuyasu Spreading Function

The Mitsuyasu sea surface spreading can be written compactly as [80]

$$D(f, \theta) = \frac{\Gamma(s+1)}{2\sqrt{\pi}\Gamma(s+1/2)} \left[\cos^2\left(\frac{\theta - \theta_0}{2}\right)\right]^s \tag{A.17}$$

where $\Gamma(s)$ represents the Gamma function; the parameter s controls the angular distribution of energy along frequency as follows:

$$s = \begin{cases} 9.77\,(f/f_m)^{-2.5} & f \geq f_m \,, \\ 6.97\,(f/f_m)^{5} & f < f_m \,; \end{cases} \tag{A.18}$$

the definition of s reflects the increase of the parameter near the peak spectral frequency and its decrease at low frequencies.

Hasselmann Spreading Function

The Hasselmann sea surface spreading can be written compactly as [75, 81]

$$D(f, \theta) = \frac{1}{N_p}\left[\cos^2\left(\frac{\theta - \theta_0}{2}\right)\right]^p \tag{A.19}$$

Through a maximum likelihood technique collected data was fitted to analytical expressions in order to obtain the following dependence of the parameters on frequency:

$$p = 9.77\left(\frac{f}{f_m}\right)^{\mu} \tag{A.20}$$

and

$$\mu = \begin{cases} 4.06 & f < f_m \,, \\ -2.34 & f > f_m \,; \end{cases} \tag{A.21}$$

The normalization factor corresponds to

$$N_p = 2^{1-2p}\pi \frac{\Gamma\left(2p+1\right)}{\Gamma^2\left(p+1\right)} \tag{A.22}$$

where $\Gamma(p)$ stands again for the Gamma function.

A.3 Swells

A swell is a long-wavelength surface wave, which is far more stable in this direction and frequency than normal wind-induced surface waves. Swells are often created by storms thousands of nautical miles away from the areas where they are observed. Such large distances allow the waves comprising the swells to become more stable and clean as they travel towards the coast. The level of energy contained in swells is influenced by the following factors: wind velocity, wind area (i.e. the amount of ocean surface area which is affected by wind blowing in the same direction, also known as fetch) and wind duration. During a typical open ocean winter storm wind speed can achieve up to 23.15 m/s, blowing over 1000 km km for 36 h will produce a periodic swell with a characteristic period of 17–20 s. Swell characteristics can be predicted using ocean surface models as the NOAA (National Oceanic and Atmospheric Administration) ocean surface model WAVEWATCH III.

Appendix B
Solving a System of Linear Equations

Many problems of computational physics are equivalent to a problem of Linear Algebra, in which one searches the solution of a system of linear equations, given by

$$\mathbf{y} = \mathbf{E}\mathbf{x} \tag{B.1}$$

where $\mathbf{y}$ is a vector of size $N \times 1$, $\mathbf{x}$ is a vector of size $L \times 1$ and $\mathbf{E}$ is a matrix of size $N \times L$. The system can be *overdetermined*, meaning that $N > L$ (the system contains more equations than unknowns) or *undetermined*, meaning that $N < L$ (the system contains less equations than unknowns). In any case, the solution $\hat{\mathbf{x}}$ is such that $\mathbf{n} = \mathbf{y} - \mathbf{E}\hat{\mathbf{x}} \neq \mathbf{0}$, and $\mathbf{n}$ is sometimes called the *noise* vector. The solution of Eq. (B.1) for each case can be described as follows:

- If the system is overdetermined ($N > L$) we can consider a solution $\hat{\mathbf{x}}$, which minimizes the square of the norm of the noise vector, given by

$$\mathbf{n}^T\mathbf{n} = (\mathbf{y} - \mathbf{E}\mathbf{x})^T(\mathbf{y} - \mathbf{E}\mathbf{x}) = \left(\mathbf{y}^T - \mathbf{x}^T\mathbf{E}^T\right)(\mathbf{y} - \mathbf{E}\mathbf{x})$$

  When $\mathbf{x} = \hat{\mathbf{x}}$ one would have that

$$\left.\frac{\partial\left(\mathbf{n}^T\mathbf{n}\right)}{\partial\mathbf{x}}\right|_{\mathbf{x}=\hat{\mathbf{x}}} = \left.\frac{\partial\left(\mathbf{n}^T\mathbf{n}\right)}{\partial\mathbf{x}^T}\right|_{\mathbf{x}^T=\hat{\mathbf{x}}^T} = \mathbf{0}$$

  The partial derivatives can be written explicitly as

$$\frac{\partial\left(\mathbf{n}^T\mathbf{n}\right)}{\partial\mathbf{x}} = \left(\mathbf{y}^T - \mathbf{x}^T\mathbf{E}^T\right)(-\mathbf{E})$$

  and

$$\frac{\partial\left(\mathbf{n}^T\mathbf{n}\right)}{\partial\mathbf{x}^T} = -\mathbf{E}^T(\mathbf{y} - \mathbf{E}\mathbf{x})$$

© The Editor(s) (if applicable) and The Author(s), under exclusive license to Springer Nature Switzerland AG 2023

O. C. Rodríguez, *Fundamentals of Underwater Acoustics*,
https://doi.org/10.1007/978-3-031-31319-6

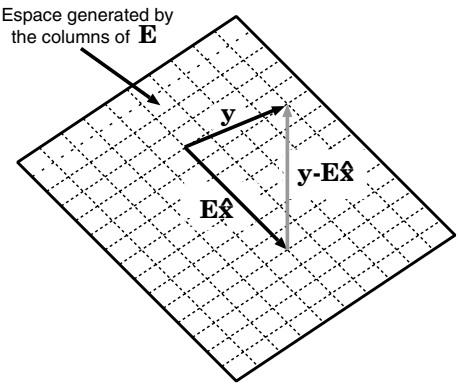


Fig. B.1 Least-squares solution: orthogonality of the noise vector to the space defined by the columns of $\mathbf{E}$ ($\text{rank}(\mathbf{E}) = L$)

It follows from the second equation that $\mathbf{E}^T\left(\mathbf{y} - \mathbf{E}\hat{\mathbf{x}}\right) = \mathbf{0}$, and thus

$$\hat{\mathbf{x}} = \left(\mathbf{E}^T\mathbf{E}\right)^{-1}\mathbf{E}^T\mathbf{y}$$

which is known as the *least-squares solution*; the existence of the inverse matrix is guaranteed by the condition that $N > L$. Additionally, the condition $\mathbf{E}^T\left(\mathbf{y} - \mathbf{E}\hat{\mathbf{x}}\right) = \mathbf{0}$ implies that the noise vector becomes orthogonal to the space defined by the columns of $\mathbf{E}$ (see Fig. B.1) [82].

- Very often the calculation of $\mathbf{E}^T\mathbf{E}$ is numerically unstable. For this case one considers the minimization of the function

$$f = \alpha\mathbf{x}^T\mathbf{x} + (\mathbf{y} - \mathbf{E}\mathbf{x})^T(\mathbf{y} - \mathbf{E}\mathbf{x})$$

where α is a "suitable" parameter. The partial derivatives for this case can be written as

$$\frac{\partial\left(\mathbf{n}^T\mathbf{n}\right)}{\partial\mathbf{x}} = \left(\mathbf{x}^T\mathbf{E}^T - \mathbf{y}^T\right)\mathbf{E} + \alpha\mathbf{x}^T$$

and

$$\frac{\partial\left(\mathbf{n}^T\mathbf{n}\right)}{\partial\mathbf{x}^T} = \mathbf{E}^T(\mathbf{E}\mathbf{x} - \mathbf{y}) + \alpha\mathbf{x}$$

which would lead to the solution given by

$$\hat{\mathbf{x}}(\alpha) = \left(\mathbf{E}^T\mathbf{E} + \alpha\mathbf{I}\right)^{-1}\mathbf{E}^T\mathbf{y}$$

where $\mathbf{I}$ is the identity matrix; a proper choice of α can fix the instability of the numerical calculation of the inverse matrix. For obvious reasons, the addition of the term $\alpha\mathbf{I}$ to $\mathbf{E}^T\mathbf{E}$ is called *diagonal overloading*. In the general case, $\hat{\mathbf{x}}(\alpha)$ approaches the least-squares solution when $\alpha \to 0$; this method of solution is known as *Tikhonov regularization*.

- If the system is undetermined ($N < L$) one can apply the method of Lagrange multipliers and minimize the function given by [82]

$$f = \mathbf{x}^T\mathbf{x} + (\mathbf{y} - \mathbf{E}\mathbf{x})^T\boldsymbol{\lambda} + \boldsymbol{\lambda}^T(\mathbf{y} - \mathbf{E}\mathbf{x})$$

  In this case we would have that

$$\frac{\partial(\mathbf{n}^T\mathbf{n})}{\partial\mathbf{x}} = \mathbf{x}^T - \boldsymbol{\lambda}^T\mathbf{E}$$

  and

$$\frac{\partial(\mathbf{n}^T\mathbf{n})}{\partial\mathbf{x}^T} = \mathbf{x} - \mathbf{E}^T\boldsymbol{\lambda}$$

  Proceeding as in the previous case and eliminating $\boldsymbol{\lambda}$ would lead us to the solution given by

$$\hat{\mathbf{x}} = \mathbf{E}^T\left(\mathbf{E}\mathbf{E}^T\right)^{-1}\mathbf{y}$$

  which is known as the *minimal norm solution*; it is valid as long as rank($\mathbf{E}$) $= N$, but the uniqueness of the solution cannot be guaranteed.
- In cases of rank deficiency of the matrix $\mathbf{E}$ (rank($\mathbf{E}$) $< \min(L, N)$) the more efficient way to obtain a solution is to calculate its pseudo-inverse through singular value decomposition [48].

References

1. Urick, R.J. 1983. *Principles of underwater sound*. New York: McGraw-Hill.
2. Brekhovskikh, L.M., and Yu.P. Lysanov. 2003. *Fundamentals of ocean acoustics*. New York: Springer.
3. Jensen, F., W. Kuperman, M. Porter, and H. Schmidt. 1994. *Computational ocean acoustics*. New York: AIP series in modern acoustics and signal processing.
4. Hutchinson, R. 2001. *Submarines, war beneath the waves: From 1776 to the present day*. London, Great Britain: Harper Collins.
5. Etter, P.C. 1996. *Underwater acoustic modeling*, 2nd ed. London, UK: Elsevier Science Publishers Ltd.
6. Buckingham M.J. 1991. Ocean acoustic propagation models. Technical Report EUR 13810, Commission of the European Communities.
7. Field R.V. 2008. Stochastic models: Theory and simulation. Technical report, SANDIA REPORT SAND2008-1365, Albuquerque, USA.
8. Porter, M., and A. Tolstoy. 1994. The matched field processing benchmark problems. *Journal of Computational Acoustics* 2 (3): 161–184.
9. Ziemer, R.E., W.H. Tranter, and D.R. Fannin. 1998. *Signals and systems: Continuous and discrete*, 4th ed. New Jersey: Prentice Hall.
10. Jesus S.M. 2020. Acoustic pressure and particle motion power spectrum estimation with Matlab. Technical report, Universidade do Algarve, Faro, Portugal.
11. Jesus S.M. 2008. Fundamentos de Telecomunicações I. Technical report, Universidade do Algarve, Faro, Portugal.
12. Henrique, L. 2007. *Acústica musical*. Lisboa, Portugal: Fundação Calouste Gulbenkian.
13. Rodríguez, O.C. 2002. *ATOMS project: Internal report*. CINTAL internal report, Faro, Portugal: Technical report.
14. Munk, W.H., R.C. Spindel, A. Baggeroer, and T.G. Birdsall. 1994. The heard island feasibility test. *Journal of the Acoustical Society of America* 96 (4): 2330–2342.
15. Margrave, G.F. 2003. *Numerical methods of exploration seismology with algorithms in MATLAB. Technical report*. Calgary, Canada: The University of Calgary.
16. Etter, D.M. 1997. *Engineering problem solving with MATLAB*. New York, USA: Prentice Hall.
17. Proakis, J.G., M. Salehi, and G. Bauch. 2011. *Contemporary communication systems using MATLAB*. Stamford, USA: CENGAGE Learning.
18. Jesus, S.M. 2008. *Sistemas e Sinais. Technical report*. Faro, Portugal: Universidade do Algarve.

© The Editor(s) (if applicable) and The Author(s), under exclusive license to Springer Nature Switzerland AG 2023

O. C. Rodríguez, *Fundamentals of Underwater Acoustics*,
https://doi.org/10.1007/978-3-031-31319-6

19. Spindel R.C. 1985. Adaptive methods in underwater acoustics. In *Acoustics conference*, 687–710. Reidel Publishing Company.
20. Tucholski, E.J. 2006. *Underwater acoustics and sonar, Technical report*. USA: U.S. Naval Academy.
21. Oppenheim, A.V., and R.W. Schafer. 2004. From frequency to quefrency: A history of the cepstrum. *IEEE Signal Processing Magazine* 21 (5): 95–106 September.
22. Francois, R.E., Garrison, R. 1982. Sound absorption based on ocean measurements. Part I: Pure water and magnesium sulfate contributions. *Journal of the Acoustical Society of America* 72 (1): 896–907.
23. Francois, R.E., Garrison, R. 1982. Sound absorption based on ocean measurements. Part II: Boric acid contribution and equation for total absorption. *Journal of the Acoustical Society of America* 72 (1): 1879–1890.
24. Yang, J., and T.K. Sarkar. 2005. Doppler-invariant property of hyperbolic frequency modulated waveforms. *Microwave and Optical Technology Letters* 48 (6): 1174–1179.
25. Ewing, W.M., W.S. Jardetzky, and F. Press. 1957. *Elastic waves in layered media*. USA: McGraw-Hill.
26. Schmidt, H. 1987. SAFARI, Seismo–Acoustic fast field algorithm for range—independent enviroments. User's guide. Technical report. La Spezia, Italy: Saclant Undersea Research Centre (SM-113).
27. Medwin, H., and C.S. Clay. 1998. *Fundamentals of acoustical oceanography*. San Diego, California: Academic Press Inc.
28. Coppens, A.B., and J.V. Sanders. 1976. *An introduction to the SONAR equations with applications, Technical report*. Monterey, USA: Naval Postgraduate School.
29. Porter, M. 1991. *The KRAKEN normal mode program*, Technical report. San Bartolomeo, Italy: Saclant Undersea Research (memorandum).
30. Medwin, H. 1975. Speed of sound in water for realistic parameters. *Journal of the Acoustical Society of America* 58: 1318–1324.
31. Papadakis, P.J., Taroudakis, M.I., Papadakis, J.S. 1994. Recovery of the properties of an elastic bottom using reflection coefficient measurements. In *Proceedings of the 2nd. European conference on underwater acoustics*, vol. II, 943–948. Copenhagen, Denmark.
32. Tindle, C.T. 2017. *Ocean acoustics. Technical report*. Australia: University of Auckland.
33. Hodges, R.P. 2010. *Underwater acoustics: Analysis. Design and performance of SONAR*. West Sussex, United Kingdom: Wiley.
34. Wenz, G.M. 1962. Acoustic ambient noise in the ocean: Spectra and sources. *Journal of the Acoustical Society of America* 34 (12): 1936–1956.
35. Thompson, S.R. 2009. Sound propagation considerations for a deep-ocean acoustic network. Master's thesis, Naval Postgraduate School, Monterey, CA 93943-5000. Approved for public release; distribution is unlimited.
36. Santos, P.J.M. 2012. *Ocean parameter estimation with high-frequency signals using a vector sensor array*. Ph.D. thesis, Universidade do Algarve.
37. Santos, P., Felisberto, P., Zabel, F., Jesus, S.M., Sebastião, L. 2017. Testing of the dual accelerometer vector sensor mounted on an autonomous underwater vehicle. In *Proceedings of the 4th underwater acoustics conference and exhibition*, 455–462. Skiathos, Greece.
38. Jesus, S.M., F.C. Xavier, R.P. Vio, J. Osowsky, M.V. Simoes, and E.B.F. Netto. 2020. Particle motion measurements near a rocky shore off Cabo Frio Island. *Journal of the Acoustical Society of America* 147: 4009–4019.
39. Santos, P., Felisberto, P., Zabel, F., Jesus, S.M., Matos, A. 2018. Dual Accelerometer vector sensor (DAVS)—REX'17 Sea trial data report. Technical report. Faro, Portugal: SiPLAB, University of Algarve.
40. Pinto, N., Jesus, S.M. 2020. Calibration of a dual accelerometer vector sensor (DAVS2) at the Alfeite arsenal tank. Technical report. Faro, Portugal: SiPLAB, University of Algarve.
41. Rodríguez, O.C. 2011. The TRACEO ray tracing program. SENSOCEAN Internal Report 1, SiPLAB, Campus de Gambelas, 800 FARO-PORTUGAL.

42. Rodríguez, O.C., Collis, J., Simpson, H., Ey, E., Schneiderwind, J., Felisberto, P. 2012. Seismo-acoustic ray model benchmarking against experimental tank data. *Journal of the Acoustical Society of America* 132 (2).
43. Rodríguez, O.C., F. Sturm, P. Petrov, and M. Porter. 2017. *Three-dimensional model benchmarking for cross-slope wedge propagation*. Boston, USA: In JASA proceedings of meetings on acoustics.
44. Calazan, R.M. 2018. *Numerical enhancements and parallel GPU implementation of the TRACEO3D model*. Ph.D. thesis, University of Algarve.
45. Calazan, R.M., and O.C. Rodríguez. 2018. Simplex based three-dimensional eigenray search for underwater predictions. *Journal of the Acoustical Society of America* 143 (4): 2059–2065 April.
46. Gerstoft, P. 1999. *SAGA User manual 3.0: An inversion software package*. La Spezia, Italy: Saclant Undersea Research Centre.
47. Schmidt, H. 2017. OASES Version 3.1 user guide and reference manual. Technical report. Boston, USA: Massachusetts Institute of Technology.
48. Rodríguez, O.C., Jesus, S. 2000. Physical limitations of travel-time-based shallow water tomography. *Journal of the Acoustical Society of America* 6 (108).
49. Hovem, J.M. 2013. *Ray trace modeling of underwater sound propagation, Technical report*. Trondheim, Norway: Norwegian University of Science and Technology.
50. Einstein, P.A. 1975. Underwater sonic ray tracing in three-dimensions. *Journal of Sound and Vibration* 4 (42): 503–508 May.
51. Beron-Vera, F.J., M.G. Brown, S. Tomsovic, A.L. Virovlyansky, M.A. Wolfson, and G.M. Zaslavsky. 2003. Ray dynamics in a long-range acoustic propagation experiment. *Journal of the Acoustical Society of America* 114 (1): 123–130.
52. Červený, V. 2001. *Seismic ray theory*. Cambridge UK: Cambridge University Press.
53. Popov, M.M. 2002. Ray theory and Gaussian beam method for geophysicists, Technical report EDUFBA, Universidade Federal da Bahia.
54. Wilmut. 2006. Inversion of lloyd mirror field for source track information. In *Acoustics 2006*, 523–527. New Zealand: Chirstchurch.
55. Butkov, E. 1988. *Física matemática*. Rio de Janeiro: Editora Guanabara.
56. Chapman, D.M.F., and D.D. Ellis. 1983. The group velocity of normal modes. *Journal of the Acoustical Society of America* 74 (3): 973–979 September.
57. Aluko, O. 2004. Implementation and application of dispersion-based waveguide models for shallow-water SONAR processing. Master's thesis, University of Pittsburgh, Pittsburgh, PA 15260 USA.
58. Bonnel, J., A. Thode, D. Wright, and R. Chapman. 2020. Nonlinear time-warping made simple: A step-by-step tutorial on underwater acoustic modal separation with a single hydrophone. *Journal of the Acoustical Society of America* 147 (3): 1897–1926.
59. Aki, K. 2009. *Quantitative seismology*. California, USA: University Science Books.
60. Pimentel, F.M.M. 1991. *Computation of acoustic normal modes in the ocean using asymptotic expansion methods*. Ph.D. thesis, Naval Postgraduate School, Monterey, USA.
61. Tolstoy, I. 1987. *Ocean acoustics: Theory and experiment in underwater sound*. Ed. by the Acoustical Society of America, New York.
62. Tindle, C.T. 1979. The equivalence of bottom loss and mode attenuation per cycle in underwater acoustics. *Journal of the Acoustical Society of America* 66 (1): 250–256 July.
63. Collins, M.D. 1993. The adiabatic mode parabolic equation. *Journal of the Acoustical Society of America* 94 (4): 2269–2278 October.
64. Claerbout, J.F. 1985. *Fundamentals of geophysical data processing with applications to petroleum prospecting*. Blackwell Scientific Publications, 2nd ed.
65. Lord, G.E., and T.D. Plemons. 1978. Characterization and simulation of underwater acoustics signals reflected from the sea surface. *Journal of the Acoustical Society of America* 63 (2): 378–384 February.
66. Applied Physics Laboratory University of Washington. APL-UW High-Frequency Ocean Environmental Acoustic Models Handbook. Technical Report APL-UW TR 9407, APL, 1994.

67. Jackson, D.R. 2000. *High-frequency bistatic scattering model for elastic seafloors*. Applied Physics Laboratory: Technical report.
68. Blondel, P. 2009. *The handbook of sidescan sonar. Springer Praxis Books*. Germany: Berlin.
69. Ainslie, M. 2010. *Principles of sonar performance modeling*. Berlin, Germany: Springer Praxis Books.
70. Rodríguez, O.C., A.J. Silva, A.P. Hughes, A.C. Moreira, J. Monteiro, et al. 2020. Underwater sonar as a ray tracing problem. In *INCREaSE 2019*, 255–264. Portugal: Faro.
71. Barroso, V. 2018. *Localização de Fonte Sonoras em Águas Rasas com uso de um único Arranjo de Hidrofones em Linha por meio de Modelagem Acústica Tridimensional*. Ph.D. thesis, UFRJ
72. Yang, T.C. 1987. A method of range and depth estimation by modal decomposition. *Journal of the Acoustical Society of America* 5 (82): 1736–1745.
73. Rodríguez, O.C. 2000. *Application of ocean acoustic tomography to the estimation of internal tides on the continental platform*. Ph.D. thesis, Faculdade de Ciências e Tecnologia–Universidade do Algarve (FCT-UALG), Faro, Portugal.
74. Marin, F. 2014. *Tomografia Oceânica por Tempo de Percurso Modal em Águas Rasas*. Ph.D. thesis, UFRJ.
75. Mastin, G.A., Watterberg, P.A., Mared, J.F. 1987. Fourier synthesis of ocean scenes. *IEEE Computer Graphics and Applications* 16–23.
76. Linnet, L.M., Clarke, S.J., Calder, B.R., Rzhanov, Y. 1997. The generation of a time correlated 2D random process for ocean wave motion. In *IPA97 IEEE conference*, 623–625.
77. Heitsenrether, R.M., and M. Badiey. 2004. Modeling acoustic signal fluctuations induced by sea surface roughness. *In Proceedings of the high frequency ocean acoustics conference, AIP conference proceedings* 728: 214–221.
78. CE. Coastal Engineering Technical Note. Technical note, U.S. Army Engineer Waterways Experimental Station, Coastal Engineering Research Center, P.O. Box 631 Vicksburg, Missisipi 39180, 1985.
79. Chen-Fen Huang. 1998. *Acoustic wave scattering from rough sea surface and seabed*. Ph.D. thesis, National Sun Yat-Sen University, Taiwan.
80. Mitsuyasu, H., and U. Tsuyoshi. 1989. A comparison of observed and calculated directional wave spectra in the east China sea. *Journal of the Oceanographical Society of Japan* 45: 338–349 September.
81. Hasselmann, D.E., M. Dunckel, and J.A. Ewing. 1980. Directional wave spectra observed during JONSWAP 1973. *Journal of Physical Oceanography* 10: 1264–1280 August.
82. Menke, W. 1989. *Geophysical data analysis: Discrete inverse theory*. San Diego, California: Academic Press Inc.